Comparing metrical typologies; Serial versus Parallel Optimality Theory, and gradient versus categorical constraints

Tom Lamers – 3114147

Supervisor: René Kager

Second reader: Tom Lentz

Utrecht University

Linguistics: The Study of the Language Faculty

Table of contents

1. Introduction	5
2. Background	7
2.1 Serial and Parallel Optimality Theory	7
2.1.1 Parallel OT	8
2.1.2 Serial OT/Harmonic Serialism	9
2.1.3 The differences between Parallel and Serial OT	14
2.1.4 Summary	22
2.2 Gradient Alignment and categorical alignment	23
2.2.1 The difference between gradient and categorical	23
2.2.2 The shift from gradient to categorical	25
2.3 Internally Layered Feet and new constraints	35
3. Factorial Typologies	43
3.1 Method	43
3.1.1 OTSoft	43
3.1.2 OTHelp	44
3.1.3 Comparing constraint rankings for Serial and Parallel OT.	45
3.1.4 The omission of certain constraints in tableaux	46
3.2 Results	47
3.3 The factorial typologies arranged into rhythmic categories	48
3.3.1 Summary	72
3.4 Duplicates	72
3.5 Summary	73
4. Discussion	75
4.1 Category I	75
4.2 Undergeneration	76
4.2.1 Edge-undergeneration	76
4.2.2 Consistent/persistent generation	78
4.2.3 Parallel OT	81
4.2.4 Interim summary	81
4.3 Overgeneration	81
5. Conclusions	87
Bibliography	89

Chapter 1. Introduction

This thesis will take a look at different typologies that can be created by Optimality Theory (OT). By using different assumptions about the components of OT, different typologies can emerge.

Prince & Smolensky (1993) introduced Optimality Theory and constraints for OT. The focus in the following literature was on one particular kind of OT – the kind where all candidates are evaluated at once: Parallel OT. However, as McCarthy (2010) and Pruitt (2010) show, another kind of OT is possible: one where a subset candidates are evaluated, changed, and new candidates are evaluated again until no further changes are necessary – Serial OT (or Harmonic Serialism). Furthermore, the constraints introduced by Prince & Smolensky (1993) were a mix of categorical constraints, giving at most one violation mark per single infraction of the constraint, and gradient constraints, which can potentially give more than one single violation mark per infraction. Gradient constraints were adopted in the literature, but have been critiqued since McCarthy (2003).

Martínez-Paricio & Kager (to appear) introduce a new set of constraints governing word stress, foot distribution and foot form to account for ternary stress. Besides the constraint set also introduced was the Internally Layered Ternary (ILT) foot, consisting of a disyllabic core foot with a one-foot dependent, resulting in a three-syllable foot. The typology resulting from the new Internally Layered Ternary foot and the new constraints is also presented in this paper. However, only one typology was given – a typology generated with Parallel Optimality Theory -, while with different assumptions about constraints and Optimality Theory) multiple typologies can be generated. In this thesis I will generate and discuss and discuss these additional typologies.

In this thesis, two major points of discussion within OT literature will be discussed: the difference between Parallel OT and Serial OT; and the difference between gradient constraints and categorical constraints. Furthermore, based on those discussion topics, a set of typologies concerning stress assignment in quantity-insensitive languages will be generated. This thesis will present the frameworks for generating factorial typologies that take the aforementioned discussion points as variables. The table in (1) shows the nature of the frameworks. In this thesis I will explore in what way these different versions of Optimality Theory, specifically the Serial and Parallel versions and the gradient and categorical constraints, and any interactions between them, shape the factorial typology.

(1) Factorial typologies to be discussed in this thesis

Parallel OT, gradient constraints	Parallel OT, categorical constraints
Serial OT, gradient constraints	Serial OT, categorical constraints

The first major discussion point is the difference between Parallel OT and Serial OT. Where Parallel OT evaluates every possible candidate at once, Serial OT evaluates in multiple passes of evaluation, each set of candidates generated from the winner of the last pass. A winning candidate in Parallel OT might not be generated under Serial OT, resulting in different outputs, and thus a different typology.

The second major discussion going on concerns the nature of OT constraints. Many constraints are taken to be gradient, meaning that, for instance, if one element is a distance of four away from another element, four violation marks are assigned for a constraints requiring those two elements to be as close to each other as possible. Categorical constraints work differently and will assign at most one violation mark if the two elements are not aligned, regardless of distance; or no violation marks if they are aligned.

This thesis will investigate which of the typologies mentioned in (1) is most suitable for metrical phonology by creating different typologies and comparing them. The most suitable method will generate all stress patterns found in natural languages without undergenerating any, or pathologically overgenerating additional stress patterns. I hope this thesis will contribute to the ongoing discussions about the different versions of OT, as it will undoubtedly show new strengths and weaknesses for both systems in light of a factorial typology, something which has been missing from the literature. The same goes for the discussion on constraint types, and possible interaction between OT types and constraint types.

The typologies are created with the help of computer software: OTSoft for Parallel typologies and OTHelp for Serial typologies. The resulting stress patterns are then grouped together by their rhythmic structures: unary, binary or ternary feet – or a mix -, whether one foot or multiple feet were built, etcetera. These rhythmic structures are paired with natural languages to see which methods generate (and undergenerate) which languages, and to show where overgeneration occurs.

Chapter 2 consists of a literature overview for Parallel OT and Serial OT; gradient and categorical constraints; and the Internally Layered foot and the new constraints.

The typologies resulting from the discrepancies between Serial OT and Parallel OT on the one hand and the difference between gradient constraints and categorical constraints on the other hand are presented in chapter 3.

Chapter 4 consists of a discussion of the results presented in chapter 3, and chapter 5 contains concluding remarks.

Chapter 2. Literature review

This chapter provides some background for the forthcoming analysis. Section 2.1 focusses on the differences between Serial OT and Parallel OT. Section 2.2 delves deeper into the difference between gradient and categorical constraints. Finally, section 2.3 provides some background on Internally Layered Ternary (ILT) feet, and introduces the constraint set that will be used to generate the typologies.

2.1 Serial and Parallel Optimality Theory

In the literature two distinct versions of Optimality Theory exist. The most prominent version is called Parallel OT (or Harmonic Parallelism), the other is Serial OT (or Harmonic Serialism). Below, both versions are presented.

First, some basic facts about what a grammar is according to Optimality Theory (Prince & Smolensky 1993). The machinery of OT consists of three universal components: a set of constraints (CON), a generator (GEN) and an evaluation (EVAL). Each of these plays a role in producing the output for any given input.

The generator takes the input and compiles a list of all possible outputs called candidates. CON takes the form of a set of violable constraints. These constraints are ranked in a specific order in strata for a specific language: language A can have constraints 1 dominating constraint 2, while they may be in the same stratum for Language B. Constraint 2 might dominate constraint 1 for Language C. The evaluation component, finally, selects the optimal candidate according to the constraint ranking out of the candidates. The process is usually displayed in a tableau like (2). The way the constraints are ranked defines the shape of the output.

(2) An example of an Optimality Theory tableau

/Input/	Constraint 1	Constraint 2	Constraint 3
a. Candidate A	*!		*
b. ☞Candidate B			**
c. Candidate C		*!	*

In tableau (2), the violation marks, *, show how often or how much this constraint is violated. EVAL chooses the candidate according to a last-man-standing principle. For the tableau in (2), this means that Candidate A is no longer considered after evaluating the first constraint, since Candidates B and C incur fewer violation marks. Candidates B and C are still in the running to become the eventual output when Constraint 2 is considered. At Constraint 2, Candidate C receives a violation mark while B does not, thus forcing Candidate C is out of the running, leaving Candidate B as the winning candidate. Candidate B is considered the output. The fact that Candidate A, like B, does not violate Constraint 2 does not matter, nor does the fact that Candidate B violates Constraint 3 more than the other candidates. Once a winner is selected evaluation stops, and a discarded candidate can never enter the competition again.

A different constraint ranking would have produced a different output. For instance, Candidate A would be the winner if the constraint ranking was Constraint 3 >> Constraint 2 >> Constraint 1. A specific constraint ranking can be referred to as a grammar, as it provides a way to map inputs to outputs.

More constraints mean more possible rankings, which in turn produces more possible grammars. The amount of possible grammars is the faculty of the number of constraints: *n*!. Note that not all of these grammars will produce different languages – some of them will produce identical outputs.

2.1.1 Parallel OT

In much of the literature concerning Optimality Theory (Prince & Smolensky, 1993), a GEN is adopted which generates every single logically possible output to be considered by EVAL. This means that for an input of /a/, the set of candidates is infinite, containing [z] and [ba], and even more outlandish concoctions as [bababababa] and [igsgiosdog]. In fact, the candidate set generated by GEN is the same for every input. The number of structural changes to the input is infinite. All possible candidates are evaluated, and after one round of evaluations, the winner becomes the output. There is a single-step mapping from input to output. A typical parallel OT table looks like (3). A high vowel has the feature [+High], a low vowel has the feature [+Low] and a mid vowel has neither [+High] nor [+Low]. Note that, while all logically possible candidates are evaluated, only the most relevant constraints are actually shown as candidates.

(3) An example of a Parallel Optimality Theory tableau

/cice/	*HighVowel	*MidVowel	Id(Low)	*LowVowel	Id(High)	Id(Mid)	*FinalVowel
a. cice	*!	*					*
b. cici	*!*				*	*	*
c. cica	*!		*	*			*
d. cece		*!*			*	*	*
e. ceca		*!	*	*	*	**	*
f. ceci	*!	*			**	**	*
g. cace		*!	*	*	*		*
h. caca			**!	**	*	*	*
i. caci	*!		*	*	**	*	*
☞j. cac			*	*	*	*	
k. cec		*!			*	**	
l. cic	*!					*	
m. cacac			**!	**	*	*	

Because a two markedness constraints (*HighVowel, *MidVowel) are ranked above a faithfulness constraint (Id(Low)), the candidate which has undergone structural changes that satisfy the markedness constraints becomes the winner. Candidate J, [cac] emerges as output because the high vowel /i/ from the input has been changed to the low vowel /a/. The markedness constraint *LowVowel is also present, but ranked lower than *MidVowel and *HighVowel, leading to a preference for low vowels. Both *MidVowel and *HighVowel are also ranked above Id(Mid) and Id(High), two faithfulness constraints which aim to preserve mid and high vowels respectively. The mid vowel /e/ has been deleted to satisfy *FinalVowel, thereby avoiding a violation mark for *MidVowel and simultaneously pre-empting changing the vowel to a low /a/ as in candidate H, which receives two violation marks for Id(Low). Note that adding a consonant to the input would also satisfy *FinalVowel, but leaves the candidate with two vowels, leading to more violation marks than forms with only one vowel.

Because every single possible candidate (including [bababababa] and [igsgiosdog]) is evaluated, the candidate best fitting the constraint ranking always wins – the global optimum is reached in every tableau.

Because the core of this thesis is about metrical phonology, I will now give a metrical example to illustrate how this is handled in Parallel OT. In the next section, I will do the same for Serial OT. Much like the example in (3), where every possible candidate is present (though only the relevant candidates are shown), any input fed to Parallel OT will be parsed by GEN into a uniquely footed structure. EVAL will then pick the structure which proves to be the best fit for the constraint ranking. An example is given in (4), although not with every possible candidate present, as the candidate set is infinite. The FtBin constraint assigns a violation mark for any foot that is not disyllabic, while Parse-σ assigns one violation mark for each unparsed syllable. All-Ft-Left assigns one violation mark for each syllable every foot is removed from the left edge, and *Trochee forbids trochaic feet. For the sake of simplicity, no difference is made between primary and secondary stress.

(4) Metrical phonology in Parallel OT

/σσσσσ/	FtBin	Parse-σ	All-Ft-Left	*Trochee
☞a. (σ’σ)(σ’σ)σ		*	**	
b. (σ’σ)σ(‘σσ)		*	***!	*
c. (‘σ)(‘σ)(‘σσ)(‘σ)	*!**		*,**,****	*
d. σ(‘σσ)σσ		**!*	*	*
e. (‘σ)(σ’σ)σσ	*!	**	*	

Candidate A is the winner since it does not violate FtBin, only has one unparsed syllable (parsing this syllable into a foot would violate FtBin) and has its iambic feet as close to the left edge as possible.

In the next section Serial OT will be discussed.

2.1.2 Serial OT/Harmonic Serialism

McCarthy (2000, 2006, 2010a) proposed a theoretical framework for Harmonic Serialism (HS), also called Serial OT. In contrast to Parallel OT, HS is not always complete after one pass through GEN and EVAL. Instead of generating every possible output, GEN is more restricted and only allows for one change to the input. Every candidate resulting from one single change to the input is considered by EVAL. The winning candidate of this first round is then used as the input for the second round through GEN and EVAL. The winning candidate must always represent a harmonic improvement over the input. If this is not possible, the input becomes the winning candidate, and *convergence* happens. McCarthy (2007, 2010b) shows that the change GEN can make is limited to one unfaithful operation such as feature change, syncope, epenthesis. There is no limit to the amount of faithful operations such as resyllabification GEN can perform.

After a pass through GEN, all information about the original input is lost. In the second round, again only one change is allowed before EVAL picks the winner to be used in the next round. This continues until the output selected as winner is the same form as the input was: convergence.

The example in (5) shows an input /cice/ going through four evaluation rounds with the same constraint ranking as in (3), finally converging on [cac]. Since only one feature change at a time is allowed a high vowel can only change to a mid vowel, and a low vowel can also only change to a mid vowel. A mid vowel can become either a low or a high vowel. As shown in (3), the best candidate would be [cac] – a form without mid and high vowels. This candidate does not occur in the first pass though EVAL in (5), but becomes available after more passes.

(5) An example of multiple passes in Serial OT

/cice/	*HighVowel	*MidVowel	Id(Low)	*LowVowel	Id(High)	Id(Mid)	*FinalVowel
a. cici	**!				*	*	*
b. cica	*!		*	*		*	*
c. cice	*!	*					*
☞d. cece		**			*	*	*
e. cic	*!				*		

Pass 1

/cece/	*HighVowel	*MidVowel	Id(Low)	*LowVowel	Id(High)	Id(Mid)	*FinalVowel
f. cece		**!					*
g. cice	*!	*			*	*	*
h. ceca		*	*!	*		*	*
i. ceci	*!	*			*	*	*
j. cace		*	*!	*		*	*
☞k. cec		*			*	*	

Pass 2

/cec/	*HighVowel	*MidVowel	Id(Low)	*LowVowel	Id(High)	Id(Mid)	*FinalVowel
j. cec		*!					
☞k. cac			*	*		*	
l. cic	*!				*	*	
m. ce		*!					*

Pass 3

/cac/	*HighVowel	*MidVowel	Id(Low)	*LowVowel	Id(High)	Id(Mid)	*FinalVowel
n. cec		*!	*			*	
☞o. cac				*			
p. ca				*			*!

Pass 4

At the first pass, five candidates are evaluated. Note that at most one structural change was made to the input for each of the candidates. The winning candidate is [cece], where a high vowel is changed to a mid vowel, satisfying *MidVowel in the process. In the second pass the final vowel of [cece] is deleted to satisfy *FinalVowel and to minimize the amount of violation marks received for *MidVowel. The form [cec] wins because candidates H and J have changed a mid vowel to a low vowel, leading to a violation mark for Id(Low). In the third pass through EVAL the vowel /e/ in [cec] is changed to a low vowel /a/. Convergence happens at the fourth pass and [cac] becomes the output.

Note that candidate [caca] was not available during the first round, as it requires two structural changes from the original input [cice]. The most crucial principle of HS is that GEN only makes one single change to the input for each candidate, and the winning candidate will be the most harmonically improved candidate. Because of this, it is not always possible to reach a global optimum, which can be defined as the best possible candidate for the given constraint ranking. Instead, Serial OT always converges on a local optimum – the best candidate that can be reached through changing the input one change at a time - which is possibly also a global optimum. The difference can be observed in the example provided in McCarthy (2006): Final-C requires that every word ends in a consonant, while Coda-Cond prohibits syllable-final obstruents. When both are ranked above Max, a constraint that incurs a violation mark for every deleted segment, every word will be shortened until it ends in a non-obstruent consonant, as in (6), where /palasanataka/ is used as input for Parallel OT.

(6) /palasanataka/ in Parallel OT

/palasanataka/	Final-C	Coda-Cond	Max
palasanataka	*!		
palasanatak		*!	*
palasanata	*!		**
palasanat		*!	***
palasana	*!		****
☞palasan			*****

Crucially, the shortening displayed in (5) is not a phenomenon found in any natural language. According to McCarthy (2006) in Serial OT, all constraints must be ranked, even when they are non-conflicting (and therefore unrankable in Parallel OT). In (7), the Serial OT evaluation of /palasanataka/ is given if Coda-Cond dominates Final-C, while in (8) the reverse is shown.

(7) /palasanataka/ in Serial OT, with Coda-Cond >> Final-C

/palasanataka/	Coda-Cond	Final-C	Max
☞palasanataka		*	
palasanatak	*!		*

Pass 1

After one pass through EVAL, the winning form is identical to the input, which means it becomes the output. The only other candidate does not become the winner because removing the final vowel does not result in harmonic improvement – the deletion results in one violation mark for the dominating constraint.

The result of Final-C dominating Coda-Cond is shown in (8).

(8) /palasanataka/ in Serial OT, with Final-C >> Coda-Cond

/palasanataka/	Final-C	Coda-Cond	Max
a. palasanataka	*!		
b. ☞palasanatak		*	*

Pass 1

/palasanatak/	Final-C	Coda-Cond	Max
c. ☞palasanatak		*	
d. palasanata	*!		*

Pass 2

Convergence happens after two steps in this situation. In both situations, there is no one single change to be made to the input that will result in harmonic improvement.

Even if, against McCarthy (2006), Coda-Cond and Final-C share the same stratum, the result from Parallel OT is not replicated, as shown in (9), where conversion is instant.

(9) /palasanataka/ in Serial OT, with Final-C and Coda-Cond sharing a stratum

/palasanataka/	Final-C	Coda-Cond	Max
a. ☞palasanataka	*		
b. palasanatak		*	*!

Pass 1

In this case, Max breaks the tie after both candidates score the same amount of violation marks in the first stratum. Since Max disfavours deleting any segments, [palasanataka] is the eventual winner.

For metrical phonology, Pruitt (2010) described a model called Iterative Foot Optimization (IFO). One of the structural changed GEN is allowed to make to the input in Serial OT is the creation of a foot. With the creation of a foot also comes stress placement within that foot. IFO aims to build the best-fitting foot for the constraint ranking. At each pass, GEN produces a candidate set from the input and at most one change, thus at most one additional foot. Pruitt (2010) postulates that it is not allowed to remove feet that have already been built, or to add syllables to feet that have been built previously. The table in (10) shows an example of building an initial foot.

(10) Metrical phonology in Serial OT

/σσσσσ/	FtBin	Parse-σ	All-Ft-Left	*Trochee
a. σσσσσ		*!****		
b. (‘σ)σσσσ	*!	****		
c. σ(‘σ)σσσ	*!	****	*	
d. σσ(‘σ)σσ	*!	****	**	
e. σσσ(‘σ)σ	*!	****	***	
f. σσσσ(‘σ)	*!	****	****	
g. (‘σσ)σσσ		***		*!
h. σ(‘σσ)σσ		***	*!	*
i. σσ(‘σσ)σ		***	*!*	*
j. σσσ(‘σσ)		***	*!**	*
k.☞(σ’σ)σσσ		***		
l. σ(σ’σ)σσ		***	*!	
m. σσ(σ’σ)σ		***	*!*	
n. σσσ(σ’σ)		***	*!**	

Pass 1

The winner is the candidate H, with a disyllabic iambic foot at the left edge. Monosyllabic feet are in violation of top-ranked FtBin, thus candidates B-F are eliminated. Candidate A, unparsed and unchanged from the input, is eliminated because it violates Parse-σ more than candidates G-N. Candidates H-J and L-N with feet that are not aligned with the left edge are also eliminated because they violate All-Ft-Left. Finally, candidate G is not the winner because it has a trochaic foot, leaving candidate K to win. The second pass is shown in (11).

(11)

/(σ’σ)σσσ/	FtBin	Parse-σ	All-Ft-Left	*Trochee
a. (σ’σ)σσσ		***!		
b. (σ’σ)(‘σ)σσ	*!	**	**	
c. (σ’σ)σ(‘σ)σ	*!	**	***	
d. (σ’σ)σσ(‘σ)	*!	**	****	
e. (σ’σ)(‘σσ)σ		*	**	*!
f. (σ’σ)σ(‘σσ)		*	***!	*
☞g. (σ’σ)(σ’σ)σ		*	**	
h. (σ’σ)σ(σ’σ)		*	***!	

Pass 2

In pass 2, the input has changed from an entirely unparsed form to the winner of pass 1, in this case (σ’σ)σσσ. The unparsed form is unavailable, and so are any of the candidates that did not win in pass 1. The process in pass 2 is the same as in pass 1: GEN creates all candidates that arise from the input plus one structural change. The winning candidate is the candidate that has an iambic foot closest to the left word edge. The unchanged candidate loses again because it violates Parse-σ more than the other candidates. Candidates with monosyllabic feet are again in violation of top-ranked FtBin and candidates with feet aligned with the right edge violate All-Ft-Left more than candidates with feet closer to the left edge. Candidate G wins because candidate E has a trochaic foot, penalized by *Trochee. Candidate G becomes the input for the third pass through EVAL in (12).

(12)

/(σ’σ)(σ’σ)σ/	FtBin	Parse-σ	All-Ft-Left	*Trochee
☞a. (σ’σ)(σ’σ)σ		*	**	
b. (σ’σ)(σ’σ)(‘σ)	*!		**,****	

Pass 3

In the third pass through EVAL, only two candidates are created. Candidate A wins because candidate B has a non-binary foot. Since candidate A is unchanged from the input, convergence happens and the derivation ends. The eventual winner is the same as the winner in (4) with the same constraint ranking, but in Parallel OT.

I will now briefly reflect on the differences between Parallel OT and Serial OT, and show positive and negative aspects of both.

2.1.3 The differences between Parallel and Serial OT

Before delving into specific examples of the differences between Parallel and Serial OT, I will here name some of the general properties of both, and name some advantages and drawbacks of both.

First, Serial OT represents a highly local process, while Parallel OT is considered non-local. One is not per se better than the other, although it has been said that locality is “a principle of linguistic metatheory: rules and constraints are local, a requirement often expressed by saying that rules or constraints do not count beyond two in their definitions” (McCarthy 2003:80, Chomsky 1965, Hayes 1995, McCarthy & Prince 1986, Nelson & Toivonen 2001). An example of non-locality is shown in (6), where Parallel OT allows truncation to happen and more than one element is allowed to be deleted at one time. In the local process shown in (7), (8) and (9), per candidate only one item is targeted by change at a time. Because Parallel OT can operate non-locally, the optimal candidate for a given constraint ranking is always available in the candidate set, whereas for Serial OT there needs to be a way to derive this candidate from the input. Serial OT does not always converge on the global optimum. This is not a shortcoming of Serial OT, merely a difference.

In this section, I will show examples of the differences between Parallel and Serial OT, and how those differences have both advantages and drawbacks. The first two examples are of processes working in tandem. In the first instance, it will turn out that these processes can easily be captured by Serial but not Parallel OT, while the reverse is true for the second example. Then I will give an example of both Serial OT and Parallel OT showing typological overgeneration. These examples will, I hope, show that neither framework is strictly superior to the other.

I have shown in previous sections that Serial OT represents a gradual, local process, while Parallel OT represents a global, instant process. One of the consequences of this difference between Parallel OT and Serial OT is that the serial version has intermediate forms. Parallel OT maps input to output in one step, while Serial OT often needs more than one step. The intermediate forms can be both a help and a hindrance. I will show an example from McCarthy (2008, 2010) to show how intermediate forms help, and an example from Kager (1996) how similar intermediate forms can form complications.

First I will show the example to in favour of intermediate forms. In Macushi Carib (Hawkins 1950), words are parsed into iambic feet, after which unstressed vowels are deleted. Two examples are shown in (13), taken from McCarthy (2010).

(13) Stress-syncope interaction in Macushi Carib

Underlying form	Stress placement	Syncope	
piripi	(pirí)(pí)	(prí)(pí)	‘spindle’
wanamari	(waná)(marí)	(wná)(mrí)	‘mirror’

McCarthy (2008) shows the process in Serial OT with the constraint ranking of Parse-σ >> *V-Place$_{\text{WEAK}}$ >> Max. *V-Place$_{\text{WEAK}}$ assigns one violation mark for any vowel in the weak syllable of a foot, while Max is a constraint that penalizes deletion of any segment. Structural changes that can be made to the input include deletion of a vowel, and creating a foot. Because Parse-σ is top-ranked, creating feet and parsing syllables is the top priority. When this happens, vowels in weak syllables of those feet will violate *V-Place$_{\text{WEAK}}$. When every syllable is parsed, segments will start to be deleted. The tableau in (14) shows what happens when both feet in *wanamari* have been created.

(14) Pass 3 of /wanamari/ in Serial OT

Input: (waná)(marí)	Parse-σ	*V-Place$_{\text{WEAK}}$	Max
a. (waná)(marí)		**!	▨
☞b. (wná)(marí)		*	*
☞c. (waná)(mrí)		*	*

Pass 3.

Since candidate A has two vowels in weak syllables, that candidate will not go on to be the winner. The winner is one of candidates B and C, in both of which a vowel in a weak syllable has been deleted – additional constraints have to break the tie. In the next pass, the other vowel will be deleted, leaving (wná)(mrí) as the winner. The underlying forms has been changed into the intermediate form, which will them change into the eventual output. It is harder to capture this process in Parallel OT. The issue is not that the intended winner is not among the list of candidates in GEN – this list is said to be infinite – but rather that there is no way to assign it to be the winner. The process in Serial OT is clear: first iambic feet are created, then unstressed vowels are deleted. This local cause-and-effect path cannot be followed in a single-step mapping process. Short of introducing a constraint that encourages deleting every vowel in an odd-numbered syllable, there is no way to distinguish between (wná)(mrí) and (wán)(már) (the latter the result of parsing trochaic feet and then deleting unstressed vowels) – at least not on grounds related to the processes of stress assignment and syncope. One could argue that a NoCoda constraint can break the tie, but this constraint is (i) unrelated to the process, (ii) might predict the faulty (wá)(mí) where the codas are simply deleted and (iii) cannot decide between a forms where codas are present everywhere.

It is crucial to note that stress placement and syncope in this case do not happen simultaneously, but rather one after the other. The next example shows a case with two processes occurring simultaneously. It will be visible in this example that Parallel OT is well-equipped to handle this, while Serial OT struggles.

It has been reported that certain parts of Estonian morphology are dependent on syllable counting (Mürk 1991, Kager 1996), and a similar issue exists in Panoan (Gonzalez 2005). Syllable counting may be inherently difficult for Serial OT to represent (McCarthy 2010). After all, Serial OT is a system in which locality has a crucial role. Looking ahead, or syllable counting, is similar to the examples in (7), (8) and (9), where a non-global optimum is reached.

In Estonian, the genitive plural affix /-te/ is used for words of two and four morae, while the /-tte/is used for words with three morae (Mürk 1991). Kager (1996) shows that foot structure, rather than the number of syllables is used in analysing this phenomenon. This is a problem for Serial OT, however, as not just one domain is analysed, but two – foot structure and affixation - are analysed simultaneously. With only one single change permitted it is impossible to simultaneously optimize both foot structure and affixation. Some affixes are incorporated into previously built feet, a problem when following Pruitt (2010) who posits that in Serial OT a foot, once built, cannot be changed. As an example, here follows a tableau taken from Kager (1996), showing the process in Parallel OT. Ft-Bin means that feet must be binary, either syllabically or moraically. Align-Hd-L secures initial main stress. Align-St-R is satisfied if each right edge of the stem coincides with the right edge of a foot. Onset assigns a violation marks for each syllable without an onset. *μ Onset assigns a violation mark for each onset that is moraic (forcing syllabification of the geminate (moraic) consonant in the /-tte/ affix with the preceding stem vowel). Shown in (15) is the moraic analysis of both /-te/ and /-tte/.

(15) Moraic analysis of Estonian genitive affixes /-te/ and /-tte/

```
 μ                          μ μ
 |                          | |
t e      /-te/       and    t e      /-tte/
```

Pk-Prom stands for Peak Prominence, a constraint that prefers a rhythmic pattern in which stressed syllables are heavy. For example, [(pá.rat).ta.(màt.tus)] is more preferable than [(pá.rat).(tà.mat).tus]. One more constraint is needed: Parse-2, which requires that one of two adjacent stress units (μ, σ) must be parsed by a foot. The Parallel OT tableau for the input of /paraja, [gen.pl]/ is as in (16), taken from Kager (1996:9).

(16) /paraja,[gen.pl]/ in Parallel OT, from Kager (1996)

/paraja,[gen.pl]/	Ft-Bin	Parse-2	Align-Hd-L	Onset	*μ Onset	Align-St-R	PkProm
a.☞[(pá.ra).(jà-t.te)]						*	L,H
b.[(pá.ra).(jà-te)]						*	L,L!
c. [pa.(rá.ja)-te]			*!				L
d. [(pá.ra).ja-te]		*!				*	L
e. [(pá).(rà.ja)-te]	*!						L
f. [(pá.ra).(jà)-te]	*!						L
g. [(pá.ra.ja)-te]	*!						L

The table in (17) shows the process in Serial OT. I will assume that syllabification happens simultaneously with footing, not costing an extra pass through EVAL, as in McCarthy (2007, 2010b, Pruitt 2010). I will also assume that there is some constraint, outranked by all others, which encourages the filling of the [gen.pl], which I will call *Empty. Furthermore, I assume that an unfilled position, such as the affix here represented by [gen.pl], cannot be footed or phonologically interacted with until it has been filled. Therefore, it cannot be made into a foot on the first pass, for example.

In Serial OT, the input stays the same: /paraja,[gen.pl]/, but the process is different, as shown in (16) below.

(17) /paraja,[gen.pl]/ in Serial OT

/paraja,[gen.pl]/	Ft-Bin	Parse-2	Align-Hd-L	Onset	*µ Onset	Align-St-R	Pk-Prom	*Empty
a. pa.ra.ja,[gen.pl]		*!*				*		*
b. paraja-te		*!**				*		
c. paraja-tte		*!**				*		
d.☞ (pá.ra).ja,[gen.pl]						*	L	*
e. pa.(rá.ja),[gen.pl]			*!				L	*
f. (pá.ra.ja),[gen.pl]	*!						L	*

Pass 1

(pá.ra).ja,[gen.pl]	Ft-Bin	Parse-2	Align-Hd-L	Onset	*µ Onset	Align-St-R	Pk-Prom	*Empty
a. (pá.ra).(jà),[gen.pl]	*!						L	
b. (pá.ra).ja-te		*!				*	L	
c. (pá.ra).ja-tte		*!				*	L	
d.☞ (pá.ra).ja,[gen.pl]						*	L	*
e. (pá.ra).ja-tte		*!			*			

Pass 2

Convergence happens on the second pass. However, candidate D is the winner, with an unfilled [gen.pl] affix position. If *Empty was a top-ranked constraint, forcing the affix position to be filled (placing it as second-ranked stratum would not alter the outcome from pass 1), candidates B and C in the first pass would have both been used as input for the second pass, illustrated in (18), since they are the only two candidates that do not violate *Empty, and cannot be teased apart via the other constraints.

(18)

pa.ra.ja-te, pa.ra.ja-tte	*Empty	Ft-Bin	Parse-2	Align-Hd-L	Onset	*µ Onset	Align-St-R	Pk-Prom
a. (pá.ra)ja-te			*				*!	L
b. (pá.ra).ja-tte			*				*!	L
c. paraja-tte			**!*				*	L
d. paraja-te			**!*				*	L
e. ☞pa.(rá.ja).-tte				*				L
g. para.(ját.te)				*			*!	H
h. para.(já.te)				*			*!	L

Alternative Pass 2

In this alternative second pass through EVAL, the winner is Candidate E because the violation mark it receives for Align-Hd-L is in the same stratum as the violations Candidates A and B receive for Parse-2, while Candidate E receives no violation mark for Align-St-R. Additionally, Candidates E and F are also ruled out because of Align-St-R. Candidate E can never end up like the winner from (16), which is the actual Estonian form.

Using Parse-Syllable, giving one violation mark per unparsed syllable, instead of Parse-2 could be a solution, since it would mean Candidate E receives two violation mark for unparsed syllables in addition to the one for Align-Hd-L, resulting in a win for candidate F (because of the heavy stressed syllable). Parse-Syllable is mentioned in Kager (1996), but placed in a stratum below Pk-Prom, whereas for this theory to be effective it should be in the place of Parse-2. While that might be effective for this particular case, it would, for instance, end up competing with Ft-Bin to build a one-syllabled foot in the case of a word with odd-numbered syllables. Re-ranking the constraints by making Pk-Prom top-ranked does allow candidate G to surface as winner, but then main stress does not fall on the initial syllable as is the case in Estonian.

The proper form, (pá.ra).(jà-t.te) can only be chosen after both the correct affix is chosen, and the affix and the preceding syllable have been footed together. Serial OT by definition cannot carry out these two changes to the input at the same time, since Serial OT can only make one single change. In this case two simultaneous changes are required; otherwise the parsing of the input goes awry. Wolf (2008, 2013) proposes a theory of Optimal Interleaving to deal with the interaction between phonology and morphology.

The next example shows that both Parallel and Serial OT can suffer from overgeneration. The first example shows how Serial OT is capable of overgeneration.

A local optimum that Serial OT can produce but that cannot be produced by Parallel OT can be problematic for Serial OT if it is an entirely unattested pattern. For instance, Hyde (2012b) describes the Odd-Parity Input Problem, made up of two sub-problems: the Odd Heavy Problem and the even output problem. The problems occur for an input with an odd number of syllables when a constraint ranking results in the tendency to create disyllabic feet, shown in (19).

(19)	a. Even-parity	b. Odd-parity
	(σσ)(σσ)(σσ)	(σσ)(σσ)(σσ)σ
		(σσ)(σσ)σ(σσ)
		(σσ)σ(σσ)(σσ)
		σ(σσ)(σσ)(σσ)

There are several solutions to accommodate the unparsed syllable. For instance, adding an extra syllable or simply deleting one will turn odd-parity into even-parity. Alternatively, if one of the syllables in the odd-parity form is heavy it might warrant its own monosyllabic foot. Hyde also presents the Odd-Parity Input Problem as a case against Serial OT, since it generates unattested output patterns that Parallel OT does not generate. The undesired output patterns of a seven-syllable input into Serial OT are shown in (20). Stresses are not shown – each output pattern exists with iambic feet or trochaic feet.

(20) (σσ)(σ)(σσ)(σσ) (σσ)(σσ)(σ)(σσ)

The reason these patterns are predicted is the iterative nature of foot building in Serial OT. The constraints used by Hyde (2012b) Parse-σ, FtBin, All-Ft-L/R and PrWd-L. Parse-σ assigns one violation mark for each unparsed syllable; PrWd-L requires the left edge of the prosodic word to be aligned with the edge of some foot. All-Ft-L/R assigns one violation mark for each syllable each foot is removed from the left/right edge. Under the constraint ranking in (21) and (22), one of the patterns in (20) emerges in Serial OT, but a different pattern emerges under Parallel OT. The tableau in (21) shows the result of an input of seven syllables in Parallel OT and the tableau in (22) shows the process in Serial OT.

(21) Seven-syllable input in Parallel OT; Parse-σ >> FtBin >> All-Ft-R, PrWd-L >> All-Ft-L

Input: σσσσσσσ	Parse-σ	FtBin	All-Ft-R	PrWd-L	All-Ft-L
☞(σσ)(σσ)(σσ)(σ)		*	*****,***,*		**,****,******
(σ)(σσ)(σσ)(σσ)		*	******,***!*,**		*,***,*****
(σσ)(σσ)(σσ)σ	*!		*****,***,*		**,****
σ(σσ)(σσ)(σσ)	*!		****,**	*	*,***,*****
(σσ)(σ)(σσ)(σσ)		*	******,***!*,**		**,***,*****
(σσ)(σσ)(σ)(σσ)		*	*****,***,**!		**,****,*****
(σσ)σ(σσ)(σσ)	*!		*****,**		***,*****
(σσ)(σσ)σ(σσ)	*!		*****,***		**,*****

(22) Seven-syllable input in Serial OT; Parse-σ >> FtBin >> All-Ft-R, PrWd-L >> All-Ft-L

Input: σσσσσσσ	Parse-σ	FtBin	All-Ft-R	PrWd-L	All-Ft-L
☞(σσ)σσσσσ	*****		*****		
σσσσσ(σσ)	*****			*****	*!****
(σ)σσσσσσ	******!	*	******		
σσσσσσσ	*******!				
σσ(σσ)σσσ	*****		***	**	*!*

Pass 1

Input: (σσ)σσσσσ	Parse-σ	FtBin	All-Ft-R	PrWd-L	All-Ft-L
(σσ)σσσσσ	****!*		*****		
☞(σσ)σσσ(σσ)	***		*****		*****
(σσ)(σσ)σσσ	***		*****,*!**		**
(σσ)σ(σσ)σσ	***		*****, *!*		***

Pass 2

Input: (σσ)σσσ(σσ)	Parse-σ	FtBin	All-Ft-R	PrWd-L	All-Ft-L
(σσ)σσσ(σσ)	**!*		*****		*****
(σσ)(σσ)σ(σσ)	*		*****,***!		**,*****
☞(σσ)σ(σσ)(σσ)	*		*****,**		***,*****

Pass 3

(Pass 4 on the next page)

Input: (σσ)σ(σσ)(σσ)	Parse-σ	FtBin	All-Ft-R	PrWd-L	All-Ft-L
(σσ)σ(σσ)(σσ)	*!		*****,**		***,*****
☞(σσ)(σ)(σσ)(σσ)		*	*****,****,**		**,***,*****

Pass 4

The output from Parallel OT is attested in natural languages, while the output from Serial OT is not.

The high ranking of Parse-σ determines that every syllable must be parsed in some way, overriding FtBin by creating a one-syllabic foot. All-Ft-R and PrWd-L are in the same stratum, creating the difference between Parallel and Serial OT. The first foot created in Serial OT is at the left edge of the word, satisfying PrWd-L maximally and violating All-Ft-R maximally. An alternative is creating a foot at the right edge, satisfying All-Ft-R maximally but violating PrWd-L maximally. The tie is broken by All-Ft-L in the next stratum, which is also maximally satisfied by the left-edged foot (if PrWd-R had been ranked in place of All-Ft-L, the tie would have been broken in favour of a right-edged foot). In the second pass, with PrWd-L already satisfied, and unable to be violated again, priority goes to minimally upsetting All-Ft-R, leading to a foot built at the right edge of the word. In the next pass through EVAL, another foot is built as close to the right edge as possible for the same reasons as the pass before. In the final pass, the one leftover syllable is parsed into a foot.

For Parallel OT the only thing that matters is violating All-Ft-R as minimally as possible while parsing every syllable. This means creating the one-syllable foot at the right edge of the word, ensuring that the three non-right-edged feet are closer to the right edge than when a disyllabic foot interferes. This is shown in (21).

The reason Serial OT does not generate a unary foot in that position is because the foot in that position is created, by the constraint ranking, as the second foot, focusing on minimally violating Parse-σ and creating the biggest possible foot – in this case a disyllabic one.

An example of overgeneration by Parallel OT is given by Pruitt (2010). Wergaia (Hercus 1969, 1986) has initial main stress and secondary stresses on other odd-numbered syllables. However, the final syllable (odd- or even-numbered) is only ever stressed if it is a heavy syllable. The pattern can be described as left-to-right trochaic, where the final syllable of a word with an odd number of syllables only bears stress if it is heavy. Words with an even number of syllables are always parsed exhaustively with trochaic disyllabic feet. In the examples below, an L stands for a light syllable, an H stands for a heavy syllable.

Wergaia can be readily analysed in Serial OT with IFO, as shown in (23) and (24). The constraints are the same ones as used in the example of the Odd-Parity Input Problem above, with the addition of the *Clash constraint, which penalizes adjacent stress-bearing syllables. It is assumed that the Trochee constraint outranks Iamb, which means only trochaic feet are considered in the examples below.

(23) An input of HLLLH, constraint ranking FtBin >> Parse-σ >> All-Ft-L >> *Clash; Serial OT

input: /HLLLL/	FtBin	Parse-σ	All-Ft-L	*Clash
a. (‘H)LLLL		****!		
☞b. (‘HL)LLL		***		
c. H(‘LL)LL		***	*!	

Pass 1

input:/(‘HL)LLL/	FtBin	Parse-σ	All-Ft-L	*Clash
a. (‘HL)LLL		**!*		
☞b. (‘HL)(,LL)L		*	**	
c. (‘HL)L(,LL)		*	***!	
d. (‘HL)(,L)LL	*!	**		

Pass 2

input:/(‘HL)(,LL)L/	FtBin	Parse-σ	All-Ft-L	*Clash
☞a. (‘HL)(,LL)L		*	**	
b. (‘HL)(,LL)(,L)	*!		**,***	

Pass 3

Due to both FtBin and Parse-σ being highly ranked, Serial OT creates binary feet that parse as many syllables as possible. While parsing a sing heavy syllable does not violate FtBin, it does violate Parse-σ more because a foot with two syllables in it is preferable and also does not violate FtBin. The final syllable is left unparsed, as paring a single light syllable would violate FtBin.

The example in (24) shows that the final syllable does get parsed if it is heavy.

(24) An input of LHHLH, constraint ranking FtBin >> Parse-σ >> All-Ft-L >> All-Ft-R; Serial OT

input: /LHHLH/	FtBin	Parse-σ	All-Ft-L	*Clash
☞a. (‘LH)HLH		***		
b. LHHL(‘H)		****!	****	
c. L(‘HH)LH		***	*!	
d. LH(‘H)LH		****!	**	

Pass 1

input:/ (‘LH)HLH /	FtBin	Parse-σ	All-Ft-L	*Clash
a. (‘LH)HLH		**!*		
b. (‘LH)(,H)LH		**	**	
c. (‘LH)(,HL)H		*	**	

Pass 2

input:/ (‘LH)(,HL)H /	FtBin	Parse-σ	All-Ft-L	*Clash
a. (‘LH)(,HL)(,H)			**,***	
b. (‘LH)(,HL)H		*!	**	

The fourth pass is not shown because no more changes can be made, so convergence must happen as there is only one candidate available. In the first two passes the initial four syllables are parsed into two disyllabic feet. The same constraint ranking in (23) and (24) ensures that the final syllable only receives stress if it is heavy – parsing it does not violate FtBin as it is dimoraic. Parallel OT can achieve the same output pattern if *Clash shares the top stratum with FtBin (Serial OT also generates the Wergaia pattern in that case). However, overgeneration appears when *Clash is ranked below All-Ft-L and an unattested partially quantity-sensitive pattern emerges. Two examples are given in (25) and (26).

(25) An input of HLLLL, constraint ranking FtBin, *Clash >> Parse-σ >> All-Ft-L >> *Clash; Parallel OT

input:/HLLLL/	FtBin	Parse-σ	All-Ft-L	*Clash
☞a. (‘H)(,LL)(,LL)			*, ***	
b. (‘HL)(,LL)L		*!	**	
c. (‘HL)(,LL)(,L)	*!		**,****	

The winning candidate has the initial heavy syllable parsed into its own foot, leaving the four light syllables parsed into two feet. The two candidates from the last pass in Serial OT are eliminated for having a non-binary foot or having parsed too few syllables.

(26) An input of LHHLH, constraint ranking FtBin, *Clash >> Parse-σ >> All-Ft-L >> *Clash; Parallel OT

input:/LHHLH/	FtBin	Parse-σ	All-Ft-L	*Clash
a. (‘L)(,HH)(,LH)	*!		*, ***	*
☞b. (‘LH)(,H)(,HL)			**, ***	*
c. (‘LH)(,HL)(,H)			**,****!	

A medial monosyllabic syllable appears if the first syllable is light and the middle syllable is heavy. The middle monosyllabic syllable places the rightmost foot closer to the left edge, thereby gaining an advantage over candidates with a middle foot consisting of two syllables. Candidate C would have won if *Clash was top-ranked with FtBin, leading to a Wergaia pattern.

Under the constraint ranking in (25) and (26), even-parity words are still parsed into trochaic feet. The odd-parity words, however, have the first heavy syllable that appears in an odd-numbered position parsed into a monosyllabic foot. This is not a pattern that is attested is natural languages, and can be described as a non-local process.

2.1.4 Summary

In the previous sections I have explained the properties of Parallel OT and Serial OT and how they differ from each other. With examples I have shown that both versions of OT have advantages and drawbacks, and that neither version is necessarily superior over the other. I have shown that both versions of OT can be positively and negatively affected when dealing with multiple processes. Similarly, both Serial OT and Parallel OT can overgenerate unattested rhythmic patterns.

In the next section, I will open a second line of inquiry, about the constraints that make up CON, one of the three elements of Optimality Theory. Constraints such as All-Ft-L/R as showcased above have been under serious discussion in Optimality Theory literature. All-Ft-L/R as presented in (19) and (20) are gradient constraints, meaning they can assign more than one violation mark per violation. The commas separating the violation marks show this: one non-aligned syllable can easily incur multiple violations. McCarthy (2003) proposed that gradient constraints should no longer be used, and that all constraints must be categorical, assigning maximally one violation mark per structure that the constraint checks. The next section will go into more detail on the debate surrounding gradient and categorical constraints, and specifically alignment constraints such as All-Ft-L/R.

2.2 Gradient Alignment and categorical alignment

In this section, the discussion concerning Gradient Alignment (GA) and the path toward Categorical Alignment (CA) is summarized.

2.2.1 The difference between gradient and categorical

OT constraints can be divided into two categories in multiple ways. One way is to group them by faithfulness and markedness, where faithfulness constraints punish any changes to the input that appear in the output, while markedness constraints punish certain patterns in the output, unrelated to the input. Another way, which will be discussed below, is to differentiate the constraints by the way they assign violation marks. This is the distinction between gradient constraints and categorical constraints, and this distinction will be further discussed in this section.

When Prince & Smolensky (1993) proposed OT, the constraint set they introduced contained both categorical and gradient constraints. An example of a categorical constraint is Onset, in (27). An example of a gradient constraint is Align(PrWd, L, Ft, L) in (28), taken from McCarthy & Prince (1993).

(27) Onset: A syllable must have an onset. Assign one violation mark for each syllable without an onset.

(28) Align(PrWd, L, Ft, L): Every left edge of the prosodic word must line up with the left edge of a foot. Assign one violation mark for each syllable between the left edge of the prosodic word and the left edge of a foot.

For (29), each syllable without an onset is in violation of the Onset constraint, and thus receives a violation. Multiple syllables can lack an onset, and thus multiple violations can be incurred for Onset. However, no single onset can incur more than one single. The table in (29) shows examples. Max is also a categorical constraint, assigning one violation mark per segment deleted from the input.

(29)

Input: VCVCV	Onset	Max
a. V.CV.CV	*!	
b. VC.VC.V	*!**	
c. ☞Ø.CV.CV		*

In this case, deleting a segment is preferred over having a syllable without an onset. In table (29), it is visible that a categorical constraint can incur more than one violation mark when it is multiply violated by different structures. McCarthy (2003) argues that a categorical constraint must adhere to the schedule in (30).

(30) *λ/C ≡ For any λ (Locus) satisfying condition C, assign a violation-mark.

The λ stands for Locus, which is the “phonological constituent that the markedness constraint militates against” (McCarthy 2003:78). In the case of Onset, the constraint can be re-defined in the schema of (30), albeit with a different wording. It is necessary, because of (30), that all constraints following this schema are formulated negatively; they penalize certain patterns rather than positively rewarding them. So, rather than ‘a syllable must have an onset’, the wording becomes ‘a syllable must not be without onset’, or, assuming that only an onset can appear between the beginning of a syllable and a nucleus, ‘a nucleus may not be at the start of a syllable’ This latter phrasing can be captured as in (31).

(31) *Nuc/ $_{\text{Syll}}$[__ : Assign one violation mark for each nucleus at the start of a syllable.

This way of describing constraints has categoricality built in. It is a yes-or-no-question: does this item λ occur in context C? If yes; assign a violation mark. If no; do nothing.

In addition to categorical constraints, there are gradient constraints, as the constraint in (28). Gradient constraints are different from categorical constraints in the sense that they can assign multiple violation marks per locus. The alignment constraint in (28) can be derived from the template in (30), taken from McCarthy & Prince (1993), by filling the empty categories.

(32) Align(Cat1, Edge1, Cat2, Edge2)

For every Cat1 there is a Cat2 such that Esge1 of Cat1 and Edge2 of Cat2 coincide.

Where Cat1, Cat2 are taken from prosodic or grammatical categories, and Edge1 and Edge2 can be left or right. (McCarthy and Prince 1993; p2)

The template in (32) comes down to asking ‘Does the edge (GEdge) of A (GCat) match the edge (PEdge) of B (PCat)?’. If yes; nothing is done. If no; violation marks are assigned. For each element *x* that occurs between the edge (L/R) of element *y* and edge (L/R) of element *z* incurs a violation. The distance is measured usually in the prosodic category beneath the lowest prosodic category mentioned in the constraint, given the hierarchy in (33).

(33) Prosodic Hierarchy (McCarthy and Prince 1993)

Prosodic Word PrWd
|
Foot Ft
|
Syllable σ

Another gradient constraint is Align(Ft, L, PrWd, L). It has the same elements as (28), but in a different order. Therefore violation marks are assigned differently. The definition is given in (34).

(34) Align(Ft, L, PrWd, L): Every left edge of every foot must line up with the left edge of the prosodic word. Assign one violation mark for each syllable between the left edge of every foot and the left edge of the prosodic word.

Instead of checking if the left edge of the prosodic word lines up with the left edge of a foot, Align(Ft, L, PrWd, L) checks the number of elements between the left edge of each foot and the left edge of the prosodic word. The tableau in (35) shows what happens when Align(Ft, L, PrWd, L) dominates Align(Ft, R, PrWd, R). The violations incurred by individual feet are given in numbers, and then added up into asterisks.

(35) Violations for gradient constraints

Input: σσσσσ	Align(Ft, L, PrWd, L)				Align(Ft, R, PrWd, R)			
☞a. (σσ)σσσ	0	-	-		3	-	-	***
b. σ(σσ)σσ	1	-	-	*!	2	-	-	**
c. σσ(σσ)σ	2	-	-	*!*	1	-	-	*
d. σσσ(σσ)	3	-	-	*!**	0	-	-	
e. (σσ)σ(σσ)	0	3	-	*!**	3	0	-	***
f. (σσ)(σσ)σ	0	2	-	*!*	3	1	-	****
g. σ(σσ)(σσ)	1	3	-	*!***	2	0	-	**
h. (σσ)(σ)(σσ)	0	2	3	*!****	3	2	0	*****

As can be seen in (35), when more feet are added the violation marks become more plentiful. Candidates A-D all receive three violation marks across two constraints. Candidate H, however, receives five violation marks for Align(Ft, L, PrWd, L). The first foot of the word does not receive any, since its left edge lines up with the left edge of the prosodic word. The second foot and third foot receive two and three violations respectively, since their left edges are remove two and three syllables from the left edge of the prosodic word.

2.2.2 The shift from gradient to categorical

In the literature, gradient constraints have been criticized. Eisner (1997) posited the question what constraints OT should allow. Under the heading of questionable constraints Align(Ft, L, PrWd, L) can be found. Eisner argues that the main problem with the family of Align-constraints is that they belong to “a family of non-local constraints that do addition”. Local constraints do not count beyond two in their definition (Chomsky 1965, Hayes 1995, McCarthy & Prince 1986, Nelson & Toivonen 2001, McCarthy 2003). As a result, GA wields unwarranted power and this power leads to unattested phenomena. The example Eisner (1997) gives, replicated in (36), concerns the constraint Align(σ, R, H, R), a constraint that needs each syllable to align with the floating tone H. The input consists of seven syllables and a yet unassigned floating tone H. The tone cannot be multiplied in this instance. Each column shows how misaligned each syllable is with the syllable containing the tone, measured in syllables.

(36)

σσσσσσσ + [H]	syllable 1	syllable 2	syllable 3	syllable 4	syllable 5	syllable 6	syllable 7	Align(σ,R,H,R)
a. σ’σσσσσσ	0	1	2	3	4	5	6	21
b. σσ’σσσσσ	1	0	1	2	3	4	5	16
c .σσσ’σσσσ	2	1	0	1	2	3	4	13
d.☞σσσσ’σσσ	3	2	1	0	1	2	3	12
e.σσσσσ’σσ	4	3	2	1	0	1	2	13
f. σσσσσσ’σ	5	4	3	2	1	0	1	16
g.σσσσσσσ’	6	5	4	3	2	1	0	21

The tone is pushed towards the middle syllable, since it only receives 12 violations there. The tone is anchored as close to the centre of the word as possible. However, if two floating tones are present, a different strategy emerges. In (37) an extra tone needs to be anchored, resulting in a different tone-placement altogether. One could expect that, like in (36), one of these tones will appear on the middle syllable, but the table in (37) shows that this is not the case.

(37)

σσσσσσσ + [H]+[H]	syllable 1	syllable 2	syllable 3	syllable 4	syllable 5	syllable 6	syllable 7	Align(σ,R,H,R)
a. σ'σσσσσσ'	0	1	2	3	2	1	0	9!
b. ☞σσ'σσσσ'σ	1	0	1	2	1	0	1	6
c. σσσ'σσ'σσ	2	1	0	1	0	1	2	7!
d. σσσ'σ'σσσ	2	1	0	0	1	2	3	9!
e. σσσσ'σ'σσ	3	2	1	0	0	1	2	9!
f. ☞σσ'σσσ'σσ	1	0	1	1	0	1	2	6

Now it is visible that for any of the winning candidates, the middle syllable does not receive a tone. As Eisner observes, the tone now wants to be anchored at ¼ and ¾ of the word. Since this word has seven syllables, the optimal positioning of either tone is not fixed, but dependent on the placement of the other tone. Eisner mentions that this sort of behaviour is non-local, and moreover not replicable in computational OT methods such as Ellison (1994) and Tesar (1996). Eisner (1996) proposes 'primitive' OT constraints, which use only temporal overlap (or lack thereof) to assign violation mark. Due to the use of a timeline, primitive constraints are inherently local – only the current point on the timeline counts without regard to the future or the past. Additionally, nothing is being measured or counted, meaning primitive constraints are also inherently categorical. Eisner (2000) shows that primitive constraints can account for stress-related instances, such as "Hayesian foot form, quantity sensitivity, simple word-initial and –final stress, iambic lengthening, directionality of footing, syllable (and foot) extrametricality, degenerate feet, and word-level stress, including prominence-based systems" (Eisner 2000:27).

McCarthy (2003) goes into great detail about the difference between gradient and categorical constraints, and is adamant that constraints must always be categorical in nature. The case presented against gradient constraints is elaborate. McCarthy describes various instances in which the use of gradient constraints either result in wrong results, or turns out to be superfluous because the same constraint can be captured by a categorical constraint with the same results. I will describe an instance of both here.

One example of getting wrong results McCarthy (2003) gives is shown in an analysis of Kager (2001), who investigates a typology with Lapse-constraints. In this instance, the use of categorical constraints prevents certain unattested from emerging (however, not every unattested pattern is eliminated). A part of the table used by McCarthy is replicated below in (38). This part of the table is only concerned with 7-syllable words that have a trochaic pattern and the main stress on the left. *Lapse assigns a violation mark for two adjacent unstressed syllables, Lapse-at-End assigns a violation mark for two adjacent syllables that are not located at the very end of the word (no additional marks for distance from the end), Lapse-at-Peak assigns a violation mark for each lapse that is not adjacent to the main stressed syllable. Align(Wd,L,Ft,L) and Align(Wd,R,Ft,R) assign a violation mark if the left/right edge of the word is not lined up with the left-right edge of a syllable. The table in (38) is unranked, meaning the violations are shown but the constraints are not in any order – it is an overview. Lapses are underlined.

(38) Unranked table showing violations for a seven-syllable input

input: σσσσσσσ	*Lapse	Lapse-at-End	Lapse-at-Peak	Align(Wd,L,Ft,L)	Align(Wd,R,Ft,R)
a. (10)(20)(20)0	*		*		*
b. (10)(20)0(20)	*	*	*		
c. (10)0(20)(20)	*	*			
d. 0(10)(20)(20)				*	

No ranking of these constraints will produce pattern b, with the fifth syllable unparsed. This is because the violations incurred by c are a proper subset of those of b – as a result c will always be more harmonious than b (or, b is said to be harmonically bound). Patterns a, c and d can still emerge and are Pintupi, Garawa and Wargamy respectively. No pattern like b has been reported in the literature. However, when a gradient constraint like Align(Ft,L,Wd,L) (each left edge of the foot must be aligned with the left edge of the word) is entered into the tableau, pattern b can emerge. The gradient constraint measures for each foot how far away from the relevant edge it is, while the categorical constraint only checks for every foot at whether or not it is located at the edge.

(39) The difference between gradient and categorical constraints

input: σσσσσσσ	Align(Ft,L,Wd,L) (gradient)	Align(Ft,L,Wd,L) (categorical)
a. (10)(20)(20)0	**,****	*,*
b. (10)(20)0(20)	**,*****	*,*
c. (10)0(20)(20)	***,*****	*,*
d. 0(10)(20)(20)	*,***,*****	*,*,*

If the constraint ranking in (40) is used, the gradient version of Align(Ft,L,Wd,L) will cause pattern b to emerge. However, if the categorical version of Align(Ft,L,Wd,L) is used, as in (41), pattern b does not emerge and is harmonically bound again.

(40)

input: σσσσσσσ	Align(Wd,R,Ft,R)	Align(Ft,L,Wd,L) (gradient)	*Lapse	Lapse-at-End	Lapse-at-Peak	Align(Wd,L,Ft,L)
a. (10)(20)(20)0	*!	**,****	*		*	
b. (10)(20)0(20)		**,*****	*	*	*	
c. (10)0(20)(20)		***,****!*	*	*		
d. 0(10)(20)(20)		*,***,***!**				*

(41)

input: σσσσσσσ	Align(Wd,R,Ft,R)	Align(Ft,L,Wd,L) (categorical)	*Lapse	Lapse-at-End	Lapse-at-Peak	Align(Wd,L,Ft,L)
a. (10)(20)(20)0	*!	*,*	*		*	
b. (10)(20)0(20)		*,*	*	*	*!	
☞c. (10)0(20)(20)		*,*	*	*		
d. 0(10)(20)(20)		*,*,*!				*

The examples in (40) and (41) shows that a gradient version of a constraint can lead to an erroneous prediction, whereas the categorical version of the same constraint does not. The categorical version of the constraint, however, is the cause of the drifting foot problem (Kager 2001, Buckley 2009), illustrated in (42) below. Each candidate has seven syllables parsed into three feet – leaving one syllable unparsed. All-Ft-L and All-Ft-R both pull one foot towards the edge of the word, leaving the middle three syllables as one foot and one unparsed syllable. This way All-Ft-L and All-Ft-R both receive two violation marks, leaving no way to differentiate between candidate A and B.

(42) Drifting foot problem, as in Martínez-Paricio & Kager (to appear)

Input: /σσσσσσσ/	All-Ft-L	All-Ft-R
A: (σ'σ)(σ,σ)σ(σ,σ)	*,*	*,*
B: (σ'σ)σ(σ,σ)(σ,σ)	*,*	*,*

With gradient constraints, there would be no tie if the one of All-Ft-L/R dominates the other – the tie is still there if they are located in the same stratum. The tableau in (43a) shows that, since the amount of syllables between the foot and the word edge matters, the order of All-Ft-L and All-Ft-R will determine whether candidate A or B is the winner. Since All-Ft-L dominates All-Ft-R, candidate A wins. (43b) shows that when the two constraints share a stratum, the drifting foot problem does occur.

(43) a. No drifting foot problem with gradient constraints

Input: /σσσσσσσ/	All-Ft-L	All-Ft-R
☞A: (σ’σ)(σ,σ)σ(σ,σ)	**,*****	***,*****
B: (σ’σ)σ(σ,σ)(σ,σ)	***,*****!	**,*****

(43) b. Drifting foot problem with gradient constraints

Input: /σσσσσσσ/	All-Ft-L	All-Ft-R
A: (σ’σ)(σ,σ)σ(σ,σ)	**,*****	***,*****
B: (σ’σ)σ(σ,σ)(σ,σ)	***,*****	**,*****

An additional constraint must break any ties. Kager (2001) uses Lapse-At-Peak to break the tie in favour of candidate B in (42) and (43b), where the lapse is located to the right of the syllable bearing main stress: (σ’σ)σ(σ,σ)(σ,σ). However, Lapse-At-Peak, or any of the Lapse constraints are not local because their locus of violation is not a single prosodic unit but a combination of two: in this case two unstressed syllables. So, using categorical constraints and local constraints does not help the drifting foot problem.

The previous examples were cases of attaining wrong results when using gradient constraints. The next example from McCarthy (2003) will show that even when gradient constraints do not get wrong results, they can be replaces with categorical alternatives, rendering gradient constraints superfluous. McCarthy (2003) shows that constraints based on the End Rule from Prince (1983) can be made categorical. There are two End Rule constraints: End Rule-L and End Rule-R. They assign marks when the head foot of a word (the foot containing the main stressed syllable) is preceded (End Rule-L) or followed (End Rule-R) by another foot. In combination with the constraint Non-Finality(Hd(Wd)), which prohibits the right edge of the final foot to coincide with the right edge of the word, a table like (44a,b) can be constructed. The language in this table has foot extrametricality, meaning that the main stress falls on the penultimate foot. The End Rule constraints in (44a) are gradient, counting the amount of syllables between the main stressed foot and the edge of the word. The violations received by a categorical interpretation of the constraints are given in (44b).

(44)a. Non-Finality and End Rule-L/R with gradient constraints

input: σσσσσσ	Non-Fin(Hd(Wd))	End Rule-R	End Rule-L
☞ a. (02)(01)(02)		**	**
b. (02)(02)(01)	*!		****
c. (01)(02)(02)		***!*	

(44)b. Non-Finality and End Rule-L/R with categorical constraints

input: σσσσσσ	Non-Fin(Hd(Wd))	End Rule-R	End Rule-L
a. (02)(01)(02)		*	*!
b. (02)(02)(01)	*!		*
☞ c. (01)(02)(02)		*	

Since there is only one foot containing main stress, the maximum amount of violations for a categorical End Rule constraint is one. It is visible that the winner resulting from gradient End Rule constraints (a) is a different winner from the winning candidate with categorical constraints (c). A categorical interpretation of the End Rule constraints can also result in a strange pattern, as the table in (45) shows, based on an example in McCarthy (2003). For an odd-syllable input – in this case seven- the main stress shifts from initial stress in to final stress.

(45) Worked-out example from McCarthy (2003)

input: σσσσσσσ	Non-Fin(Hd(Wd))	End Rule-R	End Rule-L
a (02)(01)(02)0		*!	*
☞b (02)(02)(01)0			
c (01)(02)(02)0		*!	

The emerging pattern would be one of stress on the initial foot in case of an even amount of syllables (or, the ability to parse every syllable into feet) and non-initial stress in the case of an odd amount of syllables (or, the inability to parse every syllable into feet).

A solution to this problem of extrametricality lies not in the End Rule constraints, but rather a modification of the Non-Finality constraint. Instead of not allowing the head foot of the word to share its right edge with the right edge of the word, word-final feet are prohibited (Kager 1999). The result is then as in (46).

(46)a. Gradient Non-Finality

input: σσσσσσ	Non-Fin(Ft)	End Rule-R	End Rule-L
a (02)(01)(02)	*!	** (*)	** (*)
b (02)(02)(01)	*!		**** (*)
c (01)(02)(02)	*!	**** (*)	
☞d (02)(01)00			** (*)
e (01)(02)00		*!* (*!)	

(47)b. Categorical Non-Finality

input: σσσσσσ	Non-Fin(Ft)	End Rule-R	End Rule-L
a (02)(01)(02)	*!	*	*
b (02)(02)(01)	*!		*
c (01)(02)(02)	*!	*	
☞d (02)(01)00			*
e (01)(02)00		*!	

The winner (d) has main stress on the same syllable as the winner in (44). In this case, both gradient and categorical interpretations of the End Rule constraints work and produce the same output.

The examples above show that categorical constraints either improve upon the gradient constraints resulting in a better-fitting typology, or either achieve the same result as gradient constraints, making the latter superfluous. McCarthy includes more examples, and concludes that “it is sufficient for any constraint to assign one violation-mark for each instance of the marked structure or unfaithful mapping".

In addition to the points McCarthy (2003) raises, others have insisted that gradient constraints are unwanted. Bíro (2003) shows that, in order for a constraint to be computationally tractable, it is necessary to have a linear relation between the length of an input string and the number of violations it can receive. Otherwise a constraint cannot be interpreted by a Finite State grammar. Biro (2003) notices that the maximum amount of violation marks Align(Ft, L, Wd, L) can give to an input is a quadratic function of its length. If, in an input of six syllables, all syllables are parsed into individual feet, 15 violation marks are assigned, as in (48).

(48) (σ)(σ)(σ)(σ)(σ)(σ)

0+ 1+ 2+ 3+ 4+ 5 =15

The maximum number of violation marks can be calculated with *n(n-1)/2*, with *n* being the number of syllables. The fact that the maximum number of violations is not a linear function of the number of items the constraint evaluates is shown to be computationally untraceable (Biró 2003).

Despite the arguments against gradient constraints above, Hyde (2012a) argues that distance-sensitivity in constraints is necessary. He concedes, however, that the same distance-sensitivity does yield pathological results. Instead of switching to an all-categorical constraint set, Hyde introduces a new kind of constraint: Relation-Specific Alignment (RSA). Hyde claims this constraint still adheres to the Locus of Violation from McCarthy (2003), yet manages to assign multiple violation marks per locus. Hyde presents the Two Measures Requirement in addition to the Categoricality Hypothesis in McCarthy (2003), both presented in (49).

(49) a. Two Measures Requirement: The definition of alignment constraints provides for both distance-insensitive assessment of violation marks and distance-insensitive assessment.

b. Categoricality Hypothesis: A constraint assesses no more than one violation mark per locus of violation.

While these two may seem contradictory – after all, distance–sensitive constraints are not considered categorical – but due to a different interpretation of the locus of violation this is possible. Hyde states that the locus of violation need not to be contained to one single item, but instead can be a pair (as in (50a)) or even a triplet (as in (50b)), depending on whether or not the constraint is distance-sensitive. The template for a locus of violation is given in (50). The items for which alignment is checked are named ACat, while the separator item is called SCat.

(50) a. Distance-insensitive constraint: <ACat1, ACat2>

b. Distance-sensitive constraint: <ACat1, ACat2, SCat>.

The distance-sensitive constraint in (50b) is structured so that ACat1 contains ACat2 and SCat. For instance, a constraint that checks whether a right foot edge is aligned with a right word edge and assigns a violation mark for any separating syllables has the word as ACat1, the foot as ACat2 and the syllable as SCat. This constraint will be written as $*\langle\omega, F, \sigma\rangle/[\ldots F \ldots \sigma \ldots]_\omega$. The part between arrow head shows what categories are looked at, while the part after the forward slash signifies in what way the interaction needs to occur for a violation mark. In this case, the syllable (SCat) needs to appear to the right of the foot (ACat2) within the word (ACat1) for a violation mark to be assigned. For a non-distance sensitive version, the SCat can be left out, leading to a formulation of $*\langle\omega, F\rangle/[\ldots F\ldots\sigma\ldots]_\omega$, where the syllable is still mentioned in the second part of the constraint but not the first, signifying that intervening syllables will not be counted.

The difference between (50a) and (50b) and the way violations are scored are shown in (51), two tableaux replicated from Hyde (2012a). Both constraints check whether syllables intervene between the right edge of the word and a foot. The constraint in (51b) is distance-sensitive while the constraint in (51a) is not.

(51) a. Non-distance-sensitive constraint by Hyde (2012a)

	$*\langle\omega, F\rangle/[\ldots F\ldots\sigma\ldots]_\omega$	Number of violations
a. $[\sigma_1\sigma_2\sigma_3(\sigma_4\sigma_5)_a]_A$		0
b. $[\sigma_1\sigma_2(\sigma_3\sigma_4)_a\sigma_5]_A$	$\langle\omega_A, F_\alpha\rangle$	1
c. $[\sigma_1(\sigma_2\sigma_3)_a\sigma_4\sigma_5]_A$	$\langle\omega_A, F_\alpha\rangle$	1
d. $[(\sigma_1\sigma_2)_a\sigma_3\sigma_4\sigma_5]_A$	$\langle\omega_A, F_\alpha\rangle$	1

(51) b. Distance-sensitive constraint by Hyde (2012a)

	$*\langle\omega, F,\sigma\rangle/[\ldots F\ldots\sigma\ldots]_\omega$	Number of violations
a. $[\sigma_1\sigma_2\sigma_3(\sigma_4\sigma_5)_a]_A$		0
b. $[\sigma_1\sigma_2(\sigma_3\sigma_4)_a\sigma_5]_A$	$\langle\omega_A, F_\alpha, \sigma_5\rangle$	1
c. $[\sigma_1(\sigma_2\sigma_3)_a\sigma_4\sigma_5]_A$	$\langle\omega_A, F_\alpha, \sigma_4\rangle$, $\langle\omega_A, F_\alpha, \sigma_5\rangle$	2
d. $[(\sigma_1\sigma_2)_a\sigma_3\sigma_4\sigma_5]_A$	$\langle\omega_A, F_\alpha, \sigma_3\rangle$, $\langle\omega_A, F_\alpha, \sigma_4\rangle$, $\langle\omega_A, F_\alpha, \sigma_5\rangle$	3

In (51a) the locus of violation is limited to the pair of foot F and word ω – a violation mark is assigned when a syllable σ appears to the right of a foot within the same word. Since there is only one foot, called F_a, there can only be one violation mark: either a syllable appears after the foot or not. Moving the foot further left does not impact the amount of violations, since the locus of violation does not include syllables. The pair to be investigated is always the same, since there is only one word and only one foot. This is not the case, however, for the constraint in (51b). Here, the locus of violation is extended to a triplet to include the syllable σ. The constraint in (51b) checks for each triplet consisting of the right word edge, a foot and a syllable if the syllable is situated between the right edge of the word and the foot. Since only triplets with syllables to the right of F_α receive violation marks, there is no violation in candidate a, and one in candidate b – the same as in (51a). For candidate c, however, two violation marks are assigned due to both σ_4 and σ_5 appearing to the right of F_α. Moving the foot further left does change the amount of violation marks received because the number of syllables between the right word edge and the foot increases. Since the maximum number of violation marks per locus of violation is still only one, Hyde considers this constraint to be categorical. However, the variable number of loci of violation can still cause a non-linear amount of violation marks. In that sense, the constraint is not categorical.

Hyde (2012a) proposes using constraints of both types in (50) to create bidirectional underparsing patterns. For instance, a Garawa-like pattern (Furby 1974) emerges if Foot-Left ($*\langle\omega,F\rangle/[\ldots\sigma\ldots F\ldots]_\omega$)

dominates All-Feet-Right (*<ω,F,σ>/[…σ…F…]$_ω$), as shown in (52), replicated from Hyde (2012a: p805). The Foot-Left constraint is needed to eliminate candidate d, which receive more violation marks than candidates a, b and c because none of the feet in candidate d are aligned with the left edge. This is the case in candidates a, b and c, which only receive two violation marks compared to the three violation marks for candidate d. This situation causes candidate c to emerge as winner.

(52) Garawa-like pattern from Hyde (2012a)

input: σσσσσσσ	Foot-Left, *<ω,F>/[…σ…F…]$_ω$	All-Feet-Right, *<ω,F,σ>/[…σ…F…]$_ω$
a. [(σσ)(σσ)(σσ)σ]	*,*	*****,***!,*
b. [(σσ)(σσ)σ(σσ)]	*,*	*****,***!
☞c. [(σσ)σ(σσ)(σσ)]	*,*	*****,**
d. [σ(σσ)(σσ)(σσ)]	*,*,*!	****,**

Hyde (2012a) argues that distance-sensitive constraints are still necessary. Rhythmic Licensing, while allowing an unparsed syllable near the main stress of the word, does not allow for an unparsed syllable to be between two secondary stresses, as needs to be the case in Spanish: ((grà.ma).<u>ti</u>.(cà.li).(dád)). This pattern is possible in the system Hyde proposes, as (52) shows. No distinction needs to be made between primary and secondary stress. However, it is possible that the case of Spanish above is the result of suffixation, as Hyde admits.

Another positive point for RSA constraints is that they avoid the midpoint pathology as described in Eisner (1997). With only one foot present, the non-RSA gradient constraint Align(σ, L, F, L), which checks for every syllable if its left edge is aligned with the left edge of a foot and assigns a violation marks for each intervening syllable, draws the left edge of the foot to the middle of the word to secure that as many of the other syllables are as close to it as possible, illustrated in (53).

(53) GA midpoint pathology

	Align(σ, L, F, L, σ)
a. [(σσ)σσσσσ]	*,**,***,****,***!**,******
b. [σ(σσ)σσσσ]	*,*,**,***,****,**!***
c. [σσ(σσ)σσσ]	**,*,*,**,***,****!
d. ☞[σσσ(σσ)σσ]	***,**,*,*,**,***
e. [σσσσ(σσ)σ]	****,***,**,*,*,**!
f. [σσσσσ(σσ)]	******,*****,***!*,***,**,*

An RSA constraint with the same categories can be built. To make it distance-sensitive, the separator category must be syllable σ. The other two categories are the two categories that have to line up: foot F and syllable σ. The constraint then can be formulated and shown in (54), where other constraints have to break the tie. Note that ACat2 must be contained within ACat1.

(54) All-Syllables-Left: *<F, σ, σ>/[…σ...σ...]$_F$

	*<F, σ, σ>/[…σ...σ...]$_F$
a. [(σσ)σσσσσ]	*
b. [σ(σσ)σσσσ]	*
c. [σσ(σσ)σσσ]	*
d. [σσσ(σσ)σσ]	*
e. [σσσσ(σσ)σ]	*
f. [σσσσσ(σσ)]	*

Since all feet are binary there is only one left edge of a syllable that is not aligned with the left edge of a foot within a foot. As a result all candidates receive one violation mark. It is assumed in this tableaux that a constraint on foot form dominates *<F, σ, σ>/[…σ...σ...]$_F$ because otherwise a unary foot would be the winner, as in (55), and with multiple feet a candidate composed of all unary feet might be a winning candidate, like in (56).

(55) A unary foot is the winner, showing why foot form constraints need to dominate

	*<F, σ, σ>/[…σ...σ...]$_F$
a. [(σσ)σσσσσ]	*!
b. [(σσσ)σσσσ]	*!*
c. ☞[(σ)σσσσσσ]	

(56) Another reason for foot form constraints to dominate

	*<F, σ, σ>/[…σ...σ...]$_F$
a. [(σσ)(σσ)(σσ)]	*!,*,*
b. [(σσσ)(σσσ)]	*!*,**
c. ☞[(σ)(σ)(σ)(σ)(σ)(σ)]	

Some criticisms on Hyde (2012a) are that he has found a workaround to keep gradient constraints with categorical mechanisms – which in reality do not differ from gradient constraints at all. Take the example in (52), where both Foot-Left and All-Feet-Right can be replaced by conventional non-RSA constraints. To replace Foot-Left is a categorical version of All-Ft-L, while All-Feet-Right can be replaced by a gradient version of All-Ft-Right, as in (57).

(57)

input: σσσσσσσ	Foot-Left, *<ω,F>/[…σ…F…]$_{\omega}$	All-Ft-L (categorical)	All-Feet-Right, *<ω,F,σ>/[…σ…F…]$_{\omega}$	All-Ft-Right (gradient)
a. [(σσ)(σσ)(σσ)σ]	*,*	*,*	*****,***,*	*****,***,*
b. [(σσ)(σσ)σ(σσ)]	*,*	*,*	*****,***	*****,***
c. [(σσ)σ(σσ)(σσ)]	*,*	*,*	*****,**	*****,**
d. [σ(σσ)(σσ)(σσ)]	*,*,*	*,*,*	****,**	****,**

Furthermore, besides the avoidance of the midpoint pathology, none of the issues raised in this section are solved. For instance, the computational problem Bíro (2003) raises still stands – the constraints cannot be interpreted by a Finite State Grammar (because the violation marks are identical to gradient constraints). Additionally, since the Lapse-At-Peak constraint is deemed non-local (Buckley 2009, Rice 2007) for having two adjacent syllables as locus of violation, then the proposal for RSA constraints can certainly not be considered local for having a locus of violation made up for two or three categories.

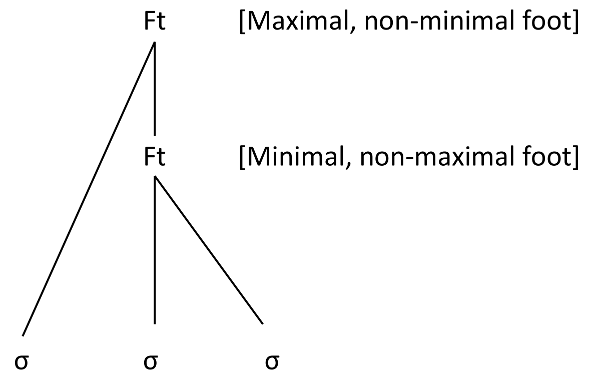
The next section will focus on Martínez-Paricio & Kager (to appear), a paper which proposes a new constraint set and foot form to account for ternary stress, while making sure the other stress systems are also being produced. Additionally, they argue to shift to local, categorical constraints instead of the global constraints associated with Gradient Alignment and Rhythmic Licensing. Section 2.3 will not delve into a new line of inquiry, but rather provide necessary background information on internally layered ternary feet and the associated constraints.

2.3 Internally Layered Ternary Feet and new constraints

The most popular way to deal with ternary rhythm in previous literature was by Weak Local Parsing (Hayes 1995), a combination of binary feet and unparsed light syllables. This kind of parsing, however, leads to pathological overgenerations such as the floating foot problem, discussed in the previous section. Martínez-Paricio & Kager (to appear) argue for the inclusion of an internally layered (ILT) foot in metrical phonology, and a new set of non-intervention constraints to better deal with ternary stress, and to eliminate pathological overgeneration. This section will explore both the ILT foot and the new constraint set.

The ILT foot is minimally recursive, resulting in two foot layers stacked on top of each other, pictured in (58). The stressed syllable is part of the foot in the lower layer, while the syllable attached to the initial foot is unstressed.

(58)



Any foot that is not part of a larger foot is considered maximal: an ILT foot consisting of a binary foot with a dependent (as in (58) above) is considered maximal (the binary foot that is part of the ILT foot is considered minimal). A binary foot not part of an ILT foot is also considered maximal, as is a unary foot.

Due to the nature of the ILT foot, four different type of ILT feet can occur, all of which are attested: (0(01)) (Winnebago), (0(10)) (Sentani), ((01)0) (Chugach Alutiiq) and ((10)0) (Cayuvava). Despite GEN's alleged Freedom of Analysis (Prince & Smolensky 1993), ternary feet have never been a staple in OT papers on metrical phonology. The Hayesian foot inventory (Hayes 1995) only includes unary and binary feet. The flat amphibrach, a ternary foot with an unstressed syllable on each side of a stressed syllable, has been covered in several papers (Levin 1985, 1988, Halle & Vergnaud 1987, Rice 1988, Halle 1990, Buckley 2009). Several papers have previously mentioned an internally layered foot (Prince 1980: 525, Selkirk 1980: 571, McCarthy 1982, Leer 1985abc, Kager 1994, 2012, Dresher & Lahiri 1991, Rice 1990, 1992, Itô &Mester 1992/2003, Hewitt 1992) in some shape or form.

The assumption that stress is the manifestation of the foot head does not change. The assumption that metrical feet are maximally binary and universally dominated by the prosodic word, however, does need to be changed. Instead, a weak syllable adjacent to a foot can be adjoined to it, creating the foot pictured in (58). Unlike Dresher and Lahiri's (1991) and Rice's (1992) layered ('resolved') foot, the innermost constituent of the ILT foot is not flat, but rather asymmetrical – meaning one of the parts of this foot is the true head of the foot, rather than the whole binary foot itself (Martínez-Patricio 2013).

The ILT foot, then, can be represented as in (58) with a minimal (and non-maximal) foot dominated by a maximal (and non-minimal) foot. A binary foot is both minimal and maximal (and neither non-minimal nor non-maximal). A foot larger than ternary, with intermediate projections between the minimal foot and the maximal foot, could of course exist if GEN has no restrictions. However, whereas there is convincing evidence for ternary feet, as laid out in Martínez-Paricio (2013), no such evidence exists for feet larger than three syllables. For instance, the maximum size of the stress window is three syllables (Kager 2012), a fact easily explained by the existence of ternary feet, yet harder to justify assuming the existence of quaternary feet or only binary feet. No language with a four-syllable stress-window has been found. Furthermore, Martínez-Paricio (2013) reviewed several prosodic foot systems and found strong support for two types of prominent syllables, and three types on non-prominent syllables. This conforms perfectly to a ternary foot structure: the two prominent syllables correspond to the head of the minimal foot and the head of the non-minimal foot, while the three kinds of non-prominent syllables are the dependent of the minimal foot, the dependent of the non-minimal foot and the adjunct of the non-minimal foot. For more arguments, see Martínez-Paricio (2013).

Locality is an important concept with regard to the constraints used in the analysis by Martínez-Paricio & Kager (to appear). They follow McCarthy (2003) in making constraints both categorical and local. Furthermore, constraints are non-interventional (Ellison 1994) and include a locus of violation (McCarthy 2003). In practice, this means a constraint as in (59) will be reinterpreted in the form of (60), resulting in (61).

(59) All-Ft-L/R: Align the left/right edge of a foot with the left/right edge of a prosodic word.

(60) Align-Left/Right (Cat1, *Cat2, Cat3): for every prosodic category Cat1, assign a violation mark if some prosodic category Cat2 intervenes between Cat1 and the left/right edge of Cat3.

(61) Align-Left/Right ([FtMax]ω, *Ft, ω): for every maximal foot $FtMax_1$, assign a violation mark if some foot intervenes between $FtMax_1$ and the left/right edge of the prosodic word.

Of course, the constraints are not identical. While (59) assigns an amount of violation marks for every foot which is not at the 'correct' edge of the prosodic word depending on how many syllables away from the edge of the word the foot is, (61) looks at every foot in the prosodic word and assigns maximally one violation mark per foot. Local, non-intervening, categorical constraints like these only check for a specific intervening element per specified phonological category and assign maximally one violation mark per instance of the specified phonological category. The difference can be seen in tables (62) and (63):

(62) Showing the violation marks for (59)

σσσσσσσσ	All-Ft-L	All-Ft-R
(σσ)(σσ)(σσ)(σσ)	**,****,******	**,****,******
(σ(σσ))(σ(σσ))(σσ)	***,******	**,*****
((σσ)σ)(σσ)((σσ)σ)	***,*****	***,*****

(63) Showing the violation marks for (61)

σσσσσσσσ	Align-Left ([FtMax]ω, *Ft, ω)	Align-Right ([FtMax]ω, *Ft, ω)
(σσ)(σσ)(σσ)(σσ)	*,*,*	*,*,*
(σ(σσ))(σ(σσ))(σσ)	*,*	*,*
((σσ)σ)(σσ)((σσ)σ)	*,*	*,*

Both binary feet and ternary feet are considered maximal feet. While the All-Ft-L/R constraints look at every foot and assign violation marks according to how far away from the edge it is, the Align-Left/Right ([FtMax]ω, *Ft, ω) inspect locally, i.e. per foot, whether or not a certain element intervenes between two pre-defined phonological categories, and assigns one violation mark per inspected foot, maximally. This results in, for instance, 12 violation marks for All-Ft-Left for an all-binary feet parse, versus three violation marks for Align-Left ([FtMax]ω, *Ft, ω). Evaluation of both constraints result in one foot that receives no violation marks (it is aligned with the left edge of the prosodic word), and three feet that do incur violation marks.

Martínez-Paricio & Kager impose restrictions on CON to limit the size of the constraint set. The Vertical Adjacency Locality Conditions consists of two restrictions CON to combat this, describing the relation between Locus of violation (Cat1), Separator category (Cat2) and Domain constituent (Cat3).

(64) Vertical Adjacency Locality Conditions (Martínez-Paricio & Kager (to appear: 8)

a. Daughterhood condition (DC): The locus of alignment (Cat1) and the separator category (Cat2) must be immediately dominated by the domain constituent (Cat3).

b. Adjacency condition (AC): The separator category (Cat2) and the domain category (Cat3) must be adjacent categories in the Prosodic Hierarchy.

With this in mind, the set of relevant constraints that emerge from this Martínez-Paricio & Kager (to appear) are shown in (65). The middle column gives an abbreviated name of the constraint.

(65) Constraints from Martínez-Paricio & Kager (to appear)

a. Align-Left/Right ([FtMax]ω, *Ft, ω)	Align-Ft-L/R	For every maximal foot FtMax, assign a violation mark if some foot intervenes between the left/right edge of its containing prosodic word.
b. Align-Left/Right ([FtMin]ω, *Ft, ω)	Align-Min-L/R	For every minimal foot FtMin, assign a violation mark if some foot intervenes between the left/right edge of its containing prosodic word.
c. Align-Left/Right ([FtNon-Min]ω, *Ft, ω)	Align-Max-L/R	For every non-minimal foot FtNon-Min, assign a violation mark if some foot intervenes between the left/right edge of its containing prosodic word.
d. Align-Left/Right ([FtUnary]ω, *Ft, ω)	Align-Un-L/R	For every unary foot FtUn, assign a violation mark if some foot intervenes between the left/right edge of its containing prosodic word.
e. Align-Left/Right ([FtMain]ω, *Ft, ω)	Align-Main-L/R	For every head of the prosodic word FtMain, assign a violation mark if some foot intervenes between the left/right edge of its containing prosodic word.
f. Align-Left/Right ([σ]ω, *Ft, ω)	Align-σ-L/R	For every unfooted syllable [σ]ω assign a violation mark if some foot intervenes between [σ]ω and the left/right edge of its containing prosodic word.
g. Align-Left/Right ([FtHead]ω, *σ, Ft)	Trochee/Iamb	For every foot head, assign a violation mark if some footed syllable intervenes between the foot head and the left/right edge of its containing foot.
h. Align-Left/Right ([FtMin]ω, *σ, Ft)	Non-MinTrochee/Non-MinIamb	For every minimal foot, assign a violation mark if some footed syllable intervenes between the minimal foot and the left/right edge of its containing foot.

To show how these constraints score violations, a series of tableaux will be given, starting with (66), which shows the constraints governing foot form and placement of main stress. The constraints are not ranked, so no winner is shown.

(66) Showing how violation marks are assigned for the constraints in (65)

Input: /σσσσσσ/	Trochee	Iamb	Non-MinTrochee	Non-MinIamb	Align-Main-L	Align-Main-R
a.σ(σ’σ)(σ(σ,σ))	*,*			*		*
b.((,σσ)σ)(σ(‘σσ))		*,*	*	*	*	
c.(σ,σ)(σ,σ)(‘σσ)	*,*	*			*	
d.σσ(σ’σσ)σ	*	*				

While Align-Left/Right([FtHead], *σ, Ft) is formulated differently than in previous literature, it works the same for the assumptions made by Martínez-Paricio & Kager (to appear) – that is, with feet of maximally two syllables (dependents are not in the foot directly containing the stressed syllable). In case of a three-syllable foot an unstressed syllable can appear to the left and the right of the foot head, resulting in a violation for both Iamb and Trochee, as in candidate D above, where an unstressed syllable is adjacent to a stressed syllable on both sides. Candidates A, B and C all receive at one violation mark per stressed syllable – either it makes the foot trochaic, violating Iamb, or I makes the foot Iambic, violating Trochee. The same goes for every ILT foot – either is it trochaic on the non-minimal level, or it is iambic. Finally, Align-Main-L/R only count feet intervening between the left or right edge of the word, meaning that candidate D does not receive any violation marks as the main foot is the only foot, its edges not coinciding with any word edges. Candidate A receives one violation mark for Align-Main-R but none for Align-Main-L, while the main foot is neither aligned with the right edge not the left edge. However, no foot intervenes between the left edge of the foot and the left edge of the word, while there is intervention between the right edges – resulting in a violation mark. The right edge of the main foot in candidate C does line up with the right edge of the word, but two feet intervene between its left edge and the left edge of the word. This intervention results in one single violation mark due to the categorical nature of the constraint.

The unranked tableau in (67) shows the working of Align-σ-L/R and Align-Ft-L/R.

(67) More constraints from (65)

Input: /σσσσσσ/	Align-σ-L	Align-σ-R	Align-Ft-L	Align-Ft-R
a.(σ,σ)σσσσ	*,*,*,*			
b.σσσ(’σσ)σ	*	*,*,*		
c.(‘σ)((,σσ)σ)σ(,σ)	*	*	*,*	*,*
d.(‘σ)σ(,σ)(,σ)σ(,σ)	*,*	*,*	*,*,*	*,*,*
e.σσσσσσ				
f.(,σ)σσσσ(‘σ)	*,*,*,*	*,*,*,*	*	*

Align-σ-L/R is formulated in such a way that unfooted syllables do not necessarily incur a violation mark, as is evident for candidate E – completely unparsed and without violation marks. When a single foot is introduced, as in candidates A and B, violation marks start pouring in. If the foot is situated at the edge, as in candidate A, violation marks for only one of Align-σ-L/R will appear. In the case of candidate A the foot is situated at the left edge, resulting in violation marks for Align-σ-L only. Since the foot intervenes between the left edge of the word and the unparsed syllables each syllable not in the foot incurs one violation mark. If, as is the case in candidate B, the foot is not situated at the edge of the word, the violation marks are spread between the left and right variant of the constraint. The first three syllables receive violation marks for Align-σ-R, since the foot intervenes between them and the right edge. The final syllable incurs a violation mark for Align-σ-L since the foot intervenes between it and the left edge. In Candidate F, the two feet situated at both edges interfere between the unfooted syllables and the left and right edges of the word, resulting in violation marks for each unfooted syllable for both Align-σ-R and Align-σ-L. Interference of multiple feet as in candidates C and D still results in maximally one violation mark per syllable.

Output candidates with a single foot do not incur violation marks for Align-Ft-L/R, since this constraint too checks for interference by feet, not unparsed syllables. The result of that fact plus the fact that this constraint is categorical is that these two constraints always have the same amount of violation marks: if foot A interferes between the left edge of the word and foot B then foot B automatically interferes between foot A and the right edge of the word.

The constraints in (68) all concern subsets of feet: non-minimal, minimal and unary feet. Binary feet are minimal feet, ILT feet are non-minimal feet and feet consisting of one syllable are unary feet and also considered minimal feet.

(68) More constraints from (65)

Input: /σσσσσσ/	Align-Min-L	Align-Min-R	Align-Max-L	Align-Max-R	Align-Un-L	Align-Un-R
a.(‘σσ)σ(σ(,σσ))		*	*			
b.(,σ)((σ’σ)σ)(σ, σ)	*	*	*	*		*
c.(‘σ)(,σ)σ((σ,σ)σ)	*	*,*	*		*	*,*

Even though the constraints all concern different kinds of feet, there is no difference between the types of feet intervening. Candidate A only incurs two violation marks in total – there are only two feet present. One minimal foot situated at the left edge and one non-minimal foot situated at the right edge. For each foot one other foot is interfering between the first foot and a word edge: the non-minimal foot interferes between the minimal foot and the right edge, resulting in a violation on Align-Min-R, while the minimal foot interferes between the left word edge and the non-minimal foot, resulting in a violation on Align-Max-L. Candidate B, with a unary foot on the left edge, has one violation on every constraint except Align-Un-L. The non-minimal foot has a foot on both sides, resulting in a violation on Align-Max-L and Align-Max-R. The minimal foot situated at the right edge of the word results in a violation of Align-Min-L. The unary foot on the left edge incurs a violation on both Align-Un-R and Align-Min-R. Since a unary foot is also considered a minimal foot the set of violations on Align-Un-L/R is always a subset of Align-Min-L/R. Candidate C has two unary feet – one on the left edge and one adjacent to it. They both incur a violation for Align-Un-R – note that the leftmost foot only incurs one violation even if there are two intervening feet, and an unparsed syllable.

Martínez-Paricio & Kager (to appear) argue that the existence of ILT feet can explain the occurrence of multiple metrically conditioned phenomena: the distribution of tones in Chugach Alutiiq; vowel lengthening in Wargamy and Yidiɲ; and the allomorphic distribution of aspirated stops and /h/ in American English. It seems sensible to use the ILT representation and the constraint set due to the typology that is provided within the paper, and the independent evidence for ILT feet outside the typology.

The research question of this thesis is: in what way do different versions of Optimality Theory, specifically the Serial and Parallel versions and the gradient and categorical constraints, and any interactions between them, shape the factorial typology? Will Serial OT, with its restricted candidate set, shrink the typology compared to Parallel OT, or will it lead to additional patterns that stem from a local optimum that Parallel OT cannot settle on? Will gradient constraints, with the additional violation marks compared to categorical, harmonically bind candidates that are not bound by their categorical counterparts, or will the lack of distance-sensitivity inherent in categorical constraints cause overgeneration? Finally, will there be an interaction effect between any of the variables? Maybe the combination of the restricted candidate set of Serial OT and the large number of violation marks from gradient constraints will cause additional undergeneration in the typology?

I will use these two variables – CON and GEN – because they are at the heart of OT, and while a lot has been written about them in previous literature, there has not been a thorough typological comparison in which different typologies have been compared. I hope this thesis will contribute to the existing literature and form the start of a discussion on the typological consequences of using different types of constraints, and different versions of OT. This can prevent a fragmented discussion on both subjects, where one author will claim a form of overgeneration here and another will claim undergeneration there, without looking at the bigger picture: how does this overgeneration influence the entire typology? Is it still a problem if there is no other overgeneration?

In the next two chapters I will compare and contrast the typologies in (1) specifically with overgeneration and undergeneration in mind. With the help of software programs output patterns from two to eight syllables are generated. Pathological overgeneration, i.e. overgeneration that leads to unnatural output patterns, is unwanted, as is the undergeneration of output patterns that are attested with languages. In the next chapter I will present these typologies.

Chapter 3. Factorial Typologies

In this chapter I will present the typologies obtained with Parallel OT and Serial OT, and categorical constraints and gradient constraints. First I will elaborate on the method I used, providing some background information on the software used: OTSoft and OTHelp. Then an overview of the rhythmic patterns resulting from the typologies is presented.

3.1 Method

Using OTSoft (Hayes, Tesar & Zuraw 2003) and OTHelp (Staubs et al. 2010), eight different factorial typologies were obtained, shown below in (69). All typologies used at least the 16 constraints listed in (65). The variables used in the input were the following: categorical (CA) or gradient (GA) constraints; the parallel or serial version of Optimality Theory; and the presence/absence of Parse-σ. Parse-σ is a constraint that assigns one violation mark for each unparsed syllable. Parallel typologies were calculated with OTSoft, serial typologies were calculated with OTHelp.

(69)

Parallel, Parse-σ, GA	Serial, Parse-σ, GA
Parallel, Parse-σ, CA	Serial, Parse-σ, CA
Parallel, no Parse-σ, GA	Serial, no Parse-σ, GA
Parallel, no Parse-σ, CA	Serial, no Parse-σ, CA

The two typologies using serial OT and without Parse-σ will not be discussed further since these constraint sets yielded only unparsed outputs, due to the lack of Parse-σ. While Align-σ-L/R eliminate the need for Parse-σ (Martínez-Paricio & Kager, to appear), this is only the case for Parallel OT. Because the input for Serial OT is entirely unparsed and built maximally one foot at a time, and Align-σ-L/R look at feet intervening between word edges and unparsed syllables this Parse-σ-eliminating effect does not occur in Serial OT.

The constraint sets including Parse-σ have 17 constraints, resulting in 17!, or 3,556,874,280,960,000 possible rankings. For the constraint set without Parse-σ, the number is slightly lower at 16!, or 20,922,789,888,000 possible rankings.

Below I will discuss in more detail the workings of OTSoft and OTHelp, and how they helped calculate the different typologies.

3.1.1 OTSoft

OTSoft is a program that helps calculate parallel OT typologies. It needs an input file with constraints, inputs, candidates and violations. It is up to the user to create this file and make sure all violation marks are assigned correctly, and that all possible candidates are included. OTSoft then ranks and re-ranks all constraints until all possible output patterns are generated.

The input file used for the parallel CA typology without Parse-σ was given to me by René Kager. It was also used for Martínez-Paricio & Kager (to appear). From Martínez-Paricio & Kager (to appear):

"Candidates to be evaluated were all logically possible metrifications of input strings ranging in length from 2 to 8 syllables. *Gen* was limited to metrifications containing all logically possible combinations of [ternary, binary and unary] feet plus unparsed syllables. We also excluded candidates of two types. First, no candidates were included whose primary stress foot occurred in a non-leftmost or non-rightmost position. Secondly, all candidates satisfied culminativity: each contained one and only one primary stress foot. Both of the excluded candidate types are harmonically bounded under the current constraint set. The total number of candidates equalled 10,612 (distributed over different word lengths). Candidates and their violations were generated by means of a computer script." (p.26)

I would add to this that candidates consisting of only unparsed syllables were also excluded.

The input files for the remaining three typologies based on this input file too: the candidate sets are identical. The violation marks for Parse-σ, as well as calculating the violation marks for the GA version of the input file were generated by means of a computer script (provided to me by my brother).

3.1.2 OTHelp

The input for OTHelp is different from the input to OTSoft, due to the differences between the serial and parallel versions of OT. OTHelp requires three files: one for the initial candidates, one for the constraints and one for the possible changes that can be made to the inputs.

The initial candidates were unparsed forms of 2 to 8 syllables for all serial typologies. The constraints were as listed in (65). The user specifies the structure that receives a violation marks, and OTSoft will assign them for each pass through EVAL. OTSoft has a function built in for the user to assign each constraint as gradient or categorical (among other options).

The input file with the allowed structural changes used Pruitt (2015) as a jumping-off point. The changes that could be made to the input were as follows. Any unstressed syllable can be made into a unary foot. Any pair of adjacent unparsed syllables can be made into a binary foot. Any binary foot with an adjacent unparsed syllable can be made into an ILT foot. This means an ILT foot needs two steps to be created – one to create a binary foot and one to add an adjacent unparsed syllable to this foot (Kager 2013). A foot that has been built cannot be undone – a removal of feet is not an option. Main stress is assigned to the first foot that is built, and can be re-assigned to other feet. A maximum of one single change can be made per candidate. Unchanged inputs are also always part of the candidate set.

The winning candidate becomes input for the next pass through EVAL. If a tie occurs, all tied candidates are used as new input. This process is repeated until the candidate consisting of unchanged input becomes the winner.

McCarthy (2006) states that in Serial OT derivation every constraint must be ranked, even when they are non-conflicting. However, OTHelp does not function this way, so for the remainder of this thesis I will follow OTHelp and leave constraints in the same stratum.

3.1.3 Comparing constraint rankings for Serial and Parallel OT.

As mentioned before, the Parse-σ constraint is not a necessary constraint for Parallel OT. It is, however, crucial to kick-start any parsing in Serial OT. After the parsing process is kick-started, Parse-σ is also no longer necessary for Serial OT. This notion is corroborated by the fact that Parse-σ can be substituted in OTHelp by a constraint that facilitates only the kick-starting of the parsing process. I will call this constraint NoParse, as in (64).

(70) NoParse: assign 10 violation marks for any candidate consisting entirely of unparsed syllables. Assign no violation marks for any candidate not consisting of entirely unparsed syllables.

If any foot is present in the candidate, NoParse assigns zero violation marks. Using the NoParse constraint in OTHelp yields the exact same typologies for Serial OT, both for the GA version and the CA version.

This constraint is not strictly necessary, but it is a great help in comparing Parallel and Serial OT, which I will do in section 3.3. In order to aid this comparison, I will take constraint rankings that produce a certain Parallel OT output and apply Serial OT to them. However, a

In section 3.3, I will compare Parallel OT and Serial OT. In order to do this, I will take constraint rankings that produce a certain Parallel OT output and apply Serial OT to them, and vice versa. A constraint ranking that produces a parsed output in Parallel OT often produces an output of only unparsed forms in Serial OT. The output pattern in (71) is generated by Parallel OT with the constraint ranking in (72).

(71) Output Pattern

(σ’σ)	(σ(σ’σ))	(‘σ)(σ(σ,σ))	(σ’σ)(σ(σ,σ))	(σ(σ’σ))(σ(σ,σ))	(‘σ)(σ(σ,σ))(σ(σ,σ))	(σ’σ)(σ(σ,σ))(σ(σ,σ))
B	**T**	**UT**	**BT**	**TT**	**UTT**	**BTT**

(72) Constraint ranking for output pattern (65), generated by OTSoft for Parallel OT

Align-σ-L, Align-σ-R, Align-Min-L, Align-Un-L, Align-FtMain-L, MaxIamb, Parse-σ >> Align-Ft-L, Align-Ft-R, Align-Max-L, Align-Max-R, Align-Min-R, Align-Unary-R, Align-FtMain-R, MinIamb, MaxTrochee >> MinTrochee

Using the first stratum of this constraint ranking and feeding it to Serial OT leads to an unparsed output, as shown in (73). Due to the presence of both Parse-σ and Align-σ-L/R any partially unparsed forms are penalized twice. Because Align-σ-L/R only assign violation marks when a form has created a foot, the unparsed form is only penalized once: only Parse-σ assigns violation marks for an entirely unparsed form.

(73) Parse-σ leads to double violation marks, leading to an unparsed output

σσσσσσσ	Align-σ-L	Align-σ-R	Align-Min-L	Align-Un-L	Align-FtMain-L	MaxIamb	Parse-σ
A: σσσσσ(σ'σ)	0	5	0	0	0	0	5!
B: (σ'σ)σσσσσ	5	0	0	0	0	0	5!
☞C: σσσσσσσ	0	0	0	0	0	0	7
D: ('σ)σσσσσσ	6	0	0	0	0	0	6!

If Parse-σ is substituted with NoParse, the winner will be a form in which parsing has taken place. This is shown in (74).

(74) NoParse instead of Parse-σ

σσσσσσσ	Align-σ-L	Align-σ-R	Align-Min-L	Align-Un-L	Align-FtMain-L	MaxIamb	NoParse
☞A: σσσσσ(σ'σ)	0	5	0	0	0	0	0
☞B: (σ'σ)σσσσσ	5	0	0	0	0	0	0
C: σσσσσσσ	0	0	0	0	0	0	10!
D: ('σ)σσσσσσ	6!	0	0	0	0	0	0

This tableau ends up in a tie. Constraints in lower strata may break the tie, or both candidates become the input for the next pass through EVAL.

Note that both Serial OT with Parse-σ and Serial OT with NoParse are able to generate the pattern in (71). The constraint rankings are shown respectively in (75) and (76).

(75) Constraint ranking for (71), Serial OT with Parse-σ

Align-Min-L, Align-Un-L, Align-σ-L, Align-FtMain-L, MaxIamb >> Parse >> Align-σ-R, Align-Ft-L, Align-Ft-R, Align-Max-L, Align-Max-R, Align-Min-R, Align-Unary-R, Align-FtMain-R, Iamb, MaxTrochee, >> Trochee

(76) Constraint ranking for (71), Serial OT with NoParse

Align-Min-L, Align-Un-L, Align-σ-L, Align-FtMain-L, MaxIamb, NoParse >> Align-σ-R >> Align-Ft-L, Align-Ft-R, Align-Min-R, Align-Max-L, Align-Max-R, Align-FtMain-R, Iamb, Align-Unary-R, MaxTrochee >> Trochee

By using NoParse instead of Parse-σ, a constraint ranking that produces a pattern in Parallel OT can be directly used with Serial OT to produce a parsed form, rather than an entirely unparsed form. There is no objection to using Serial OT constraint rankings with Parallel OT.

3.1.4 The omission of certain constraints in tableaux

In the analysis below, not every constraint will be shown in every tableau, due to space limitations. Analysis can be done without certain constraints. Firstly, constraints governing foot-form as in (59gh) will not be shown. These constraints come in pairs and, all else being equal, the dominating constraint always wins. This means that, if Trochee is ranked above Iamb, only trochaic feet will be built. With that knowledge, there is no need to include Trochee/Iamb and NonMinTrochee/NonMinIamb in the tableaux.

Also left out is Align-FtMain-L/R. These two constraints are the only two concerned with the position of main stress. Therefore, the constraint dominating the other governs where the main stress is located: either on the leftmost foot or on the rightmost foot. Since there are no interactions with other constraints, this is not an area that needs analysis.

Furthermore, due to space constraints, sometimes entire strata are left out if these strata are not involved before a winning candidate is found.

3.2 Results

Table (77) below shows the total number of distinct output patterns, qua bracketing, that is generated by each framework.

(77) Output results

Parallel CA (PCA)	316
Parallel CA with Parse (PCAP)	316
Parallel GA (PGA)	300
Parallel GA with Parse (PGAP)	428
Serial CA (SCA)	83
Serial GA (SGA)	83

As is instantly visible, Serial Optimality Theory reduces the amount of output patterns vastly compared to Parallel OT. Also instantly visible is that PCA and PCAP generate the same amount of patterns. The two are not only equal in number – there is no difference at all between PCA and PCAP. In the analysis below the output patterns are divided into rhythmic categories A-J. Without going into their characteristics now, I will present table (78) showing that PCA and PCAP are equal (and that there is little difference between PGA and PGAP). Additionally, SGA and SCA also yield equal results. In (78) it shows of each framework how many of the output patterns belong in each rhythmic category. There is no difference between PCAP and PCA, or between SGA and SCA.

(78)

	PCAP	PCA	PGAP	PGA	SCA	SGA
A	12	12	12	12	8	8
B	24	24	40	40	16	16
C	24	24	24	24	16	16
D	32	32	16	16	8	8
E	64	64	80	80	8	8
F	48	48	16	16	0	0
G	32	32	32	32	8	8
H	16	16	16	16	0	0
I	64	64	64	192	16	16
J	0	0	0	0	3	3
Total	316	316	428	300	82	82

PCA will be ignored for the remainder of the discussion. SGA and SCA also generate identical output patterns, and will often be discussed simply as Serial OT.

The next section follows the categorization as in Martínez-Paricio & Kager (to appear). This means that every foot is represented by a T, B or U (for ternary, binary, unary). These abbreviations collapse different foot forms, as laid out in (79). The forms in the Form column are used in the following output pattern tables.

(79) BTU-abbreviations

Abbreviation	Form	Collapses	
U	(σ)	(1)	(2)
B	(σσ)	(01), (10)	(02), (20)
T	(σσσ)	((10)0), ((01)0), (0(01)), (0(10))	((20)0), ((02)0), (0(02)), (0(20))

3.3 The factorial typologies arranged into rhythmic categories

First, a quick note on the following tables. The top row shows the general pattern of the output. **B(σ*)** means that one binary foot is (optionally) followed by multiple (*) unparsed syllables. Underneath the eight-syllable form are listed languages that fit the pattern, or report that the pattern is unattested. The information underneath that is the amount of patterns generated by each of the different methods.

A. Single foot systems

(780)	**B(σ*)**	**T(σ*)**
	(σσ)	(σσ)
	(σσ)σ	(σσσ)
	(σσ)σσ	(σσσ)σ
	(σσ)σσσ	(σσσ)σσ
	(σσ)σσσσ	(σσσ)σσσ
	(σσ)σσσσσ	(σσσ)σσσσ
	(σσ)σσσσσσ	(σσσ)σσσσσ
	Hungarian, Mohawk, Dakota, Turkish	Macedonian, Azkoitia Basque
PCAP	4	8
PGAP	4	8
PGA	4	8
SCA	4	4
SGA	4	4

In this category, only one main stress is assigned: there is only one foot. Due to Align-FtMain-L/R, this foot is always either on the left or the right. The binary one-foot systems generated by the different OTs are all the same: Initial stress, post-initial stress, final stress and penultimate stress. The story is slightly different for the ternary one-foot systems. Since ILT feet are generated in one step under Parallel OT and in two under Serial OT, there is a noticeable difference. For PCA and PGA, each of the four ILT feet with primary stress can occur at both the left and right edge. However, since, as is evident from the binary one-foot patterns, a single binary foot can only be a winning candidate if it is aligned with one of the edges of the prosodic word; there is always one side of the foot that does not have an adjacent unparsed syllable. This results in an inability to render half the ternary feet: hence only four ternary patterns exist. The consequence is that, under the assumption of Serial OT, there cannot be stress on the third syllable only, or the antepenultimate syllable – as is the case in Macedonian (Kager 2012). Table (81) below shows which eight-syllable **T(σ*)** outputs can be generated by Parallel OT and which can only be generated by Serial OT.

(81) Edge-undergeneration in Serial OT

Parallel OT	Serial OT
[(10)0] σσσσσ	(10) σσσσσσ -> [(10)0] σσσσσ
[(01)0] σσσσσ	(01) σσσσσσ -> [(01)0] σσσσσ
[0(10)] σσσσσ	* σ (10) σσσσσ
[0(01)] σσσσσ	* σ (01) σσσσσ
σσσσσ[0(01)]	σσσσσσ (01) -> σσσσσ[0(01)]
σσσσσ[0(10)]	σσσσσσ (10) -> σσσσσ[0(10)]
σσσσσ[(10)0]	* σσσσσ (10) σ
σσσσσ[(01)0]	* σσσσσ (01) σ

The inability of Serial OT to generate initial feet away from an edge can be called **edge-undergeneration**.

There is no difference between GA and CA patterns here.

No patterns with a single unary foot are generated.

B. Strictly binary rhythmic systems

(82)	**B*(σ)**	**B*(σ); T in 3σ**
	(σσ)	(σσ)
	(σσ)σ	(σσσ)
	(σσ)(σσ)	(σσ)(σσ)
	(σσ)(σσ)σ	(σσ)(σσ)σ
	(σσ)(σσ)(σσ)	(σσ)(σσ)(σσ)
	(σσ)(σσ)(σσ)σ	(σσ)(σσ)(σσ)σ
	(σσ)(σσ)(σσ)(σσ)	(σσ)(σσ)(σσ)(σσ)
	Pintupi, Cairene Arabic, Warao, Wargamay, Araucanian, Creek	Unattested
PCAP	8	16
PGAP	8	16
PGA	8	16
SCA	8	8
SGA	8	8

The same pattern as in the A category is repeated: Serial OT is more restrictive because ILT feet that include minimal feet not aligned with the left or right word edge (depending on which of Align-Min-L/R is dominant) cannot be generate: edge-undergeneration. In Parallel OT, the T foot in the 3σ candidate can be any of the four primary stress-bearing ILT feet, even the ones of which internal foot is not aligned with the same edge as all the other feet. This is not possible in Serial OT, which means that, in principle, the stress patterns for both B categories, with regard to stress placement, are the same for Serial OT. More on duplicate stress patterns in section 3.4.

One additional category is generated by PGA and PGAP. Like **B*(σ); T in 3σ**, it does not quite fit the B category, but it comes close.

(83)

	B*(T); Tσσ in 5σ
	(σσ)
	(σσσ)
	(σσ)(σσ)
	(σσσ)σσ
	(σσ)(σσ)(σσ)
	(σσ)(σσ)(σσ)σ
	(σσ)(σσ)(σσ)(σσ)
	Unattested
PCAP	0
PGAP	16
PGA	16
SCA	0
SGA	0

For five-syllable forms, instead of generating two binary feet and leaving one syllable unparsed, in this case one ILT foot is built, leaving two feet unparsed. The ranking in (84) generates this pattern.

(84) Align-σ-L, Align-Max-L, Align-Max-R, Align-Un-L, Align-Un-R, Align-FtMain-L, MinIamb, MaxIamb >> Align-σ-R >> Align-Ft-L, Align-Ft-R, Align-Min-L, Align-Min-R, Align-FtMain-R, MinTrochee >> MaxTrochee, Parse

(σ’σ)	(σ(σ’σ))	(σ’σ)(σ,σ)	σσ(σ(σ’σ))	(σ’σ)(σ,σ)(σ,σ)	σ(σ’σ)(σ,σ)(σ,σ)	(σ’σ)(σ,σ)(σ,σ)(σ,σ)
B	**T**	**BB**	**σσT**	**BBB**	**σBBB**	**BBBB**

The tableau in (85) below shows how this ILT foot with two unparsed syllables is generated, and why it works for PGA(P) and not for PCA.

(85) The constraint ranking from (74) in PGA and PCA

Input: σσσσσ	Align -σ-L	Align - Max-L	Align - Max-R	Align -Un-L	Align -Un-R	Align -σ-R	Align -Ft-L	Align -Ft-R	Align -Min-R	Align -Min-L
PGA1: σ(σ,σ)(σ’σ)	0	0	0	0	0	2	1!	1	1	1
☞ PGA2:σσ(σ(σ’σ))	0	0	0	0	0	2	0	0	0	0
PGA3: (σ,σ)(σ(σ’σ))	0	1!	0	0	0	0	1	1	1	0
☞PCA1: σ(σ,σ)(σ’σ)	0	0	0	0	0	1	1	1	1	1
PCA2: σσ(σ(σ’σ))	0	0	0	0	0	2!	0	0	0	0
PCA3: (σ,σ)(σ(σ’σ))	0	1!	0	0	0	0	1	1	1	0

The only difference between PGA and PCA here is the amount of violation marks assigned by Align-σ-R in PGA1 and PCA1. In the case of PGA, two violation marks are assigned because one unparsed syllable has two intervening feet. For PCA, while the same structure is present, only one violation mark is assigned. This is the result of the locus of violation: only one violation mark can be assigned per violating item.

The same constraint ranking applied to Serial OT is shown in (86). Because of the low position of Parse-σ/NoParse, the unparsed form surfaces as output.

(86) Constraint ranking in (84) Serial OT

Input: σσσσσ	Align-σ-L	Align-Max-L	Align-Max-R	Align-Un-L	Align-Un-R	Align-σ-R	Align-Ft-L	Align-Ft-R	Align-Min-R	Align-Min-L	NoParse
A: σσσ(σ’σ)	0	0	0	0	0	3!	0	0	0	0	0
B: (σ’σ)σσσ	3!	0	0	0	0	0	0	0	0	0	0
☞C: σσσσσ	0	0	0	0	0	0	0	0	0	0	10

C. Mixed binary/unary systems

(87)

	B*(U)	**B*(U); T in 3σ**
	(σσ)	(σσ)
	(σσ)(σ)	(σσσ)
	(σσ)(σσ)	(σσ)(σσ)
	(σσ)(σσ)(σ)	(σσ)(σσ)(σ)
	(σσ)(σσ)(σσ)	(σσ)(σσ)(σσ)
	(σσ)(σσ)(σσ)(σ)	(σσ)(σσ)(σσ)(σ)
	(σσ)(σσ)(σσ)(σσ)	(σσ)(σσ)(σσ)(σσ)
	Maranungku, Passamaquoddy, Ojibwa, Weri, Central Alaskan Yupik	Unattested
PCAP	8	16
PGAP	8	16
PGA	8	16
SCA	8	8
SGA	8	8

The output patterns for **B*(U)** are the same for every condition. In case of **B*(U); T in 3σ**, again Serial OT produces fewer output patterns in comparison to Parallel OT, due to the ILT foot: edge-undergeneration.

D. Mixed binary/ternary systems

(88)	**B*(T)**	**B*(T)(B)**
	(σσ)	(σσ)
	(σσσ)	(σσσ)
	(σσ)(σσ)	(σσ)(σσ)
	(σσ)(σσσ)	(σσ)(σσσ)
	(σσ)(σσ)(σσ)	(σσ)(σσ)(σσ)
	(σσ)(σσ)(σσσ)	(σσ)(σσσ)(σσ)
	(σσ)(σσ)(σσ)(σσ)	(σσ)(σσ)(σσ)(σσ)
	Pintupi, Cairene Arabic, Warao, Wargamay, Araucanian, Creek	Indonesian
PCAP	16	16
PGAP	16	0
PGA	16	0
SCA	8	0
SGA	8	0

The only difference between the two output patterns is in the 7-syllable output. Only PCA generates a **B*(T)(B)** pattern. The constraint ranking for one such ranking is as in (83).

(89) Align-σ-L, Align-σ-R, Align-Unary-L, Align-Unary-R, Align-FtMain-L, MinIamb, MaxIamb, Parse >> Align-Ft-L, Align-Ft-R, Align-Min-R, Align-FtMain-R >> Align-Max-R, Align-Min-L, MaxTrochee >> Align-Max-L

The result of this ranking for both PGA (since Parse-σ is in the first stratum and not violated, there is no difference between PGA and PGAP) and PCA is shown in (84)

(90) The constraint ranking in (89) for PCA and PGA

input: σσσσσσσ	Align-σ-L	Align-σ-R	Align-Unary-L	Align-Unary-R	Align-Ft-L	Align-Ft-R	Align-Min-R	Align-FtMain-R	Align-Max-R	Align-Min-L
☞PCA1:(01)(0(02))(02)	0	0	0	0	2	2	1	1	1	1
PCA2:(0(01))(02)(02)	0	0	0	0	2	2	1	1	1	2!
PCA3:(01)(02)(0(02))	0	0	0	0	2	2	2!	1	0	2
PGA1:(01)(0(02))(02)	0	0	0	0	3	3	2!	2	1	2
☞PGA2:(0(01))(02)(02)	0	0	0	0	3	3	1	2	2	3
PGA3:(01)(02)(0(02))	0	0	0	0	3	3	3!	2	0	1

The losing candidates for PCA are eliminated in a later stadium than the losing candidates for GA are, due to the different way of scoring violations. Align-Min-R in CA scores one violation for each minimal foot that has any foot intervening between it and the right edge, while in GA it scores one violation for each foot intervening between a minimal foot and the right edge. For the PGA-candidates, this is the decisive factor: the placement of the minimal feet is crucial. Both left, as in PGA3 results in three violations: the (01) foot incurs two and the adjacent (02) incurs one. While both PGA1 and PGA2 have a minimal foot aligned with the right edge, the leftmost minimal foot in PGA1 has two feet intervening, resulting in one more violation mark than PGA2. PCA3 is also eliminated because it has two minimal feet not aligned with the right edge, but both PCA1 and PCA2 only incur one violation for the leftmost foot. Therefore, neither is eliminated yet and the competition goes to the next stratum. Here, Align-Min-L breaks the tie in favour of PCA1: PCA1 only has one minimal foot not aligned with the left edge while PCA2 has two.

Serial OT does not generate the **B*(T)(B)** pattern either. Below in tableau (85) is shown the process for SCA. In this case, only the first stratum and the second stratum are shown, since the interactions between these constraints will adequately explain the pattern resulting from this ranking. Some outputs are also left out: only the candidates with feet generated at the edges are shown.

(91) The constraint ranking in (89) for Serial OT

input: σσσσσσσ	Align-σ-L	Align-σ-R	Align-Unary-L	Align-Unary-R	NoParse	Align-Ft-L	Align-Ft-R	Align-Min-R
☞A:(σ’σ)σσσσσ	5	0	0	0	0	0	0	0
☞B:σσσσσ(σ’σ)	0	5	0	0	0	0	0	0
C:σσσσσσ(‘σ)	0	6!	0	0	0	0	0	0
D:(‘σ)σσσσσσ	6!	0	0	0	0	0	0	0
E: σσσσσσσ	0	0	0	0	10!	0	0	0

Pass 1

On the basis of these constraints, no winner can be decided between candidates A and B. The constraints that are left out will also not break this tie, since the only relevant constraints are all in the first stratum (Align-σ-L/R and Parse-σ). Both candidate A and candidate B will be considered as input for the second pass through GEN and EVAL. If every possible candidate was shown, then all candidates with a single binary foot would have been considered. The second pass through EVAL is shown in (92).

(92) Continuation of (91)

input: (σ’σ)σσσσσ, σσσσσ(σ’σ)	Align-σ-L	Align-σ-R	Align-Unary-L	Align-Unary-R	NoParse	Align-Ft-L	Align-Ft-R	Align-Min-R
☞A:(σ’σ)(σ,σ)σσσ	3	0	0	0	0	1	1	1
☞B:σσσ(σ’σ)(σ,σ)	0	3	0	0	0	1	1	1
C:((σ’σ)σ)σσσσ	0	4!	0	0	0	0	0	0
D:σσσσ(σ(σ’σ))	4!	0	0	0	0	0	0	0
E: (σ’σ)σσσσσ	5!	0	0	0	0	0	0	0
F: σσσσσ(σ’σ)	0	5!	0	0	0	0	0	0

Pass 2

Unary feet were excluded from the first pass because binary feet parse more syllables. The same logic can be applied for ILT feet: they do not foot as many syllables as two binary feet, but take an equal amount of steps. The third pass generates an extra disyllabic foot. The fourth pass, where the last remaining syllable is footed, is shown below in (93).

(93) Continuation of (92)

input: σ(σ’σ)(σ,σ)(σ,σ), (σ’σ)(σ,σ)(σ,σ)σ	Align-σ-L	Align-σ-R	Align-Unary-L	Align-Unary-R	NoParse	Align-Ft-L	Align-Ft-R	Align-Min-R
A: (σ’σ)(σ,σ)(σ,σ)(,σ)	0	0	1!	0	0	3	3	3
B: (‘σ)(σ,σ)(σ,σ)(σ,σ)	0	0	0	1!	0	3	3	3
C:(σ’σ)(σ,σ)((σ,σ)σ)	0	0	0	0	0	2	2	2!
☞D:(σ(σ’σ))(σ,σ)(σ,σ)	0	0	0	0	0	2	2	1

The ILT foot is preferred because it does not create an additional foot: a unary foot violates Align-Unary-L/R and assigns an extra violation to Align-Ft-L/R and Align-Min-R/L. An ILT foot does neither of these, since it merely expands on an existing foot. Only at the final pass does the tie vanish. Align-Min-R favours candidate D because one minimal foot is aligned with the right edge and one is not, whereas candidate C has two minimal feet not aligned with the right edge.

The reason that BTB is not generated is stooled in locality. If an extra binary foot is judged to be the best fit for the constraint ranking once, then binary feet will be made until it is no longer possible. I will call this **consistent/persistent generation.**

The reason for the difference between Parallel OT and Serial OT for **B*(T)** is the same as outlined for the A category.

E. Mixed ternary/binary/unary systems

(94)	**T(B*)(U)**	**T*(B/U)(B)**	**T*(B/U)**	**(B)T*(U)**	**(B*)T*(U)**
	(σσ)	(σσ)	(σσ)	(σσ)	(σσ)
	(σσσ)	(σσσ)	(σσσ)	(σσσ)	(σσσ)
	(σσσ)(σ)	(σσσ)(σ)	(σσσ)(σ)	(σσσ)(σ)	(σσσ)(σ)
	(σσσ)(σσ)	(σσσ)(σσ)	(σσσ)(σσ)	(σσ)(σσσ)	(σσ)(σσσ)
	(σσσ)(σσ)(σ)	(σσσ)(σσσ)	(σσσ)(σσσ)	(σσσ)(σσσ)	(σσσ)(σσσ)
	(σσσ)(σσ)(σσ)	(σσσ)(σσ)(σσ)	(σσσ)(σσσ)(σ)	(σσσ)(σσσ)(σ)	(σσ)(σσ)(σσσ)
	(σσσ)(σσ)(σσ)(σ)	(σσσ)(σσσ)(σσ)	(σσσ)(σσσ)(σσ)	(σσ)(σσσ)(σσσ)	(σσ)(σσσ)(σσσ)
	Kashaya	Unattested	Unattested	Unattested	Unattested
PCA	16	16	16	16	0
PGAP	16	16	16	0	16
PGA	16	16	16	0	16
SCA	0	0	8	0	0
SGA	0	0	8	0	0

In the mixed ternary/binary/unary systems, only PCA can generate a **(B)T*(U)** pattern. One of these **(B)T*(U)** output patterns is the result of the ranking in (89).

(95) Align-σ-L, Align- σ-R, Align-Un-R, Align-FtMain-L, MaxTrochee, Parse >> Align-Ft-L, Align-Ft-R, Align-Min-L, Align-FtMain-R >> Align-Max-L, Align-Min-R >> Align-Max-R, Align-Un-L, MinIamb, MaxIamb >> MinTrochee

The table in (96) shows the result of this ranking for inputs of four, five, seven and eight syllables in PGAP and PCA, and the process for Serial OT (SCA) is shown in (97).

(96) The constraint ranking in (95) for PCA and PGA, 4, 5, 7 and 8 syllables

input: σσσσ	Align-Un-R	Align-Ft-L	Align-Ft-R	Align-Min-L	Align-Max-L	Align-Min-R	Align-Max-R	Align-Un-L
☞PCA1: ((σ’σ)σ)(,σ)	0	1	1	1	0	0	1	0
PCA2: (‘σ)((σ‘σ)σ)	1!	1	1	0	1	1	0	1
PCA3: (σ‘σ)(σ,σ)	0	1	1	1	0	1!	0	0
☞PGA1: ((σ’σ)σ)(,σ)	0	1	1	1	0	0	1	0
PGA2: (‘σ)((σ‘σ)σ)	1!	1	1	0	1	1	0	1
PGA3: (σ‘σ)(σ,σ)	0	1	1	1	0	1!	0	0

Input: σσσσσ	Align-Un-R	Align-Ft-L	Align-Ft-R	Align-Min-L	Align-Max-L	Align-Min-R	Align-Max-R	Align-Un-L
☞PCA1: (σ’σ)((σ,σ)σ)	0	1	1	0	1	1	0	0
PCA2: ((σ’σ)σ)(σ,σ)	0	1	1	1!	0	0	1	0
PCA3: (σ’σ)(σ,σ)(,σ)	0	2	2!	2	0	2	0	1
☞PGA1: (σ’σ)((σ,σ)σ)	0	1	1	0	1	1	0	0
PGA2: ((σ’σ)σ)(σ,σ)	0	1	1	1!	0	0	1	0
PGA3: (σ’σ)(σ,σ)(,σ)	0	3!	3	3	0	3	0	2

input: σσσσσσσ	Align-Un-R	Align-Ft-L	Align-Ft-R	Align-Min-L	Align-Max-L	Align-Min-R	Align-Max-R	Align-Un-L
☞PCA1:((σ’σ)σ)((σ,σ)σ)(,σ)	0	2	2	1	1	0	2	1
PCA2: (‘σ)((σ,σ)σ)((σ,σ)σ)	1!	2	2	0	2	1	1	0
PCA3: (σ’σ)(σ,σ)((σ,σ)σ)	0	2	2	1	1	1!	0	0
PCA4: ((σ’σ)σ)(σ,σ)(σ,σ)	0	2	2	2!	0	1	1	0
PGA1:((σ’σ)σ)((σ,σ)σ)(,σ)	0	3	3	2!	1	0	3	2
PGA2: (‘σ)((σ,σ)σ)((σ,σ)σ)	1!	3	3	0	3	2	2	0
☞PGA3: (σ’σ)(σ,σ)((σ,σ)σ)	0	3	3	1	2	3	0	0
PGA4: ((σ’σ)σ)(σ,σ)(σ,σ)	0	3	3	3!	0	1	2	0

Input: σσσσσσσσ	Align-Un-R	Align-Ft-L	Align-Ft-R	Align-Min-L	Align-Max-L	Align-Min-R	Align-Max-R	Align-Un-L
☞PCA1: (σ’σ)((σ,σ)σ)((σ,σ)σ)	0	2	2	0	2	1	1	0
PCA2: ((σ’σ)σ)((σ,σ)σ)(σ,σ)	0	2	2	1!	1	0	2	0
☞PGA1: (σ’σ)((σ,σ)σ)((σ,σ)σ)	0	3	3	0	3	2	1	0
PGA2: ((σ’σ)σ)((σ,σ)σ)(σ,σ)	0	3	3	2!	1	0	3	0

The only difference between PGAP and PCA for this constraint ranking lies in the seven-syllable form. The outcome for PCA is TTU, while the winning output for PGAP is BBT. The reason two binary feet win out over a ternary foot and a unary foot in GA but not CA is that the unary foot receives one violation mark for Align-Min-L in CA (since some foot intervenes between the unary foot and the left edge) but two violation marks in GA (because two feet interfere with the unary foot and the left edge). In CA this leads to a tie, which is decided in favour of TTU by virtue of having fewer violation marks in for Align-Min-R. In GA, there is no tie because the unary foot incurs two violation marks while the two binary feet incur only one. The result is an output pattern that only PGA and PGAP can produce. The tableau in (97) shows what SCA produces for the constraint ranking in (95) (values that differ for SGA are shown in parentheses).

(97) The constraint ranking in (95) for Serial OT, 4 syllables

Input: σσσσ	Align-σ-R	Align-σ-L	Align-Un-R	NoParse	Align-Ft-L	Align-Ft-R	Align-Min-L	Align-Max-L	Align-Min-R
☞(σ’σ)σσ	0	2	0	0	0	0	0	0	0
☞σ(σ’σ)σ	1	1	0	0	0	0	0	0	0
☞σσ(σ’σ)	2	0	0	0	0	0	0	0	0
(‘σ)σσσ	0	3!	0	0	0	0	0	0	0
σσσ(‘σ)	3!	0	0	0	0	0	0	0	0
σσσσ	0	0	0	10!	0	0	0	0	0

Pass 1

Input: (σ’σ)σσ, σ(σ’σ)σ, σσ(σ’σ)	Align-σ-R	Align-σ-L	Align-Un-R	NoParse	Align-Ft-L	Align-Ft-R	Align-Min-L	Align-Max-L	Align-Min-R
☞(σ’σ)(σ,σ)	0	0	0	0	1	1	1	0	1
σ((σ’σ)σ)	1!	0	0	0	0	0	0	0	0
(‘σ)(σ,σ)σ	0	1!(2!)	1	0	1	1	1	0	1
(σ’σ)(,σ)σ	0	1!(2!)	0	0	1	1	1	0	1
(σ’σ)σσ	0	2!	0	0	0	0	0	0	0
((σ’σ)σ)σ	0	1!	0	0	0	0	0	0	0

Pass 2

Since Parse-σ and Align-σ-L/R are all present in the first stratum, any changes that parses as many syllables as possible is going to be a winner. This results in a binary system. For inputs with an odd number of syllables, the remaining syllable becomes the dependant of an ILT foot, as the five-syllable example in (92) shows. This pattern is the same as shown in the B category: **B*(T)**

(98) The constraint ranking in (95) for Serial OT, 5 syllables

Input: (σ’σ)(σ,σ)σ, σ(σ’σ)(σ,σ)	Align-σ-R	Align-σ-L	Align-Un-R	Parse	Align-Ft-L	Align-Ft-R	Align-Min-L	Align-Max-L	Align-Min-R
☞(σ’σ)((σ,σ)σ)	0	0	0	0	1	1	0	1	1
(σ’σ)(σ, σ)(,σ)	0	0	0	0	2 (3!)	2! (3)	2 (3)	0	2 (3)
(‘σ)(σ,σ)(σ,σ)	0	1!(2!)	1(2)	0	2 (3)	2 (3)	2 (3)	0	2 (3)
(σ’σ)(σ,σ)σ	0	1!(2!)	0	1	1	1	1	0	1

Pass 3

Serial OT can also not generate **T(B*)(U)** and **T*(B/U)(B)** patterns. The constraint ranking in (99) results, for Parallel OT, in the **T(B*)(U)** pattern in (100).

(99) Align-σ-L, Align-σ-R, Align-Max-L, Align-Un-R, MaxTrochee, Align-FtMain-L, Parse >> Align-Ft-L, Align-Ft-R, Align-Min-L, Align-Min-R, Align-FtMain-R >> Align-Max-R, Align-Un-L, MinIamb, MaxTrochee >> Min Trochee

(100) The constraint ranking in (93) for Parallel OT

(σ’σ)	((σ’σ)σ)	((σ’σ)σ)(σ)	((σ’σ)σ)(σ’σ)	((σ’σ)σ)(σ,σ)(,σ)	((σ’σ)σ)(σ,σ)(σ,σ)	((σ’σ)σ)(σ,σ)(σ,σ)(,σ)
B	**T**	**TU**	**TB**	**TBU**	**TBB**	**TBBU**

Let’s see what happens when this constraint ranking is applied to Serial OT. In (95), an eight-syllable input is used.

(101) The constraint ranking in (99) for Serial OT

input: σσσσσσσσ	Align-σ-R	Align-σ-L	Align-Max-L	Align-Un-R	NoParse	Align-Ft-L	Align-Ft-R	Align-Min-L	Align-Min-R
☞A:(σ’σ)σσσσσσ	0	6	0	0	0	0	0	0	0
B: (‘σ)σσσσσσσ	0	7!	0	0	0	0	0	0	0
C: σσσσσσσσ	0	0	0	0	10!	0	0	0	0

Pass 1

input: (σ’σ)σσσσσσ	Align-σ-R	Align-σ-L	Align-Max-L	Align-Un-R	NoParse	Align-Ft-L	Align-Ft-R	Align-Min-L	Align-Min-R
A: (σ’σ)σσσσσσ	0	6!	0	0	0	0	0	0	0
B: ((σ’σ)σ)σσσσσ	0	5!	0	0	0	0	0	0	0
☞C:(σ’σ)(σ,σ)σσσσ	0	4	0	0	0	1	1	1	1
D: (σ’σ)(,σ)σσσσσ	0	5!	0	0	0	1	1	1	1

Pass 2

input: (σ’σ)(σ,σ)σσσσ	Align-σ-R	Align-σ-L	Align-Max-L	Align-Un-R	NoParse	Align-Ft-L	Align-Ft-R	Align-Min-L	Align-Min-R
A: (σ’σ)((σ,σ)σ)σσσ	0	3!	1	0	0	1	1	0	1
☞B: (σ’σ)(σ,σ)(σ,σ)σσ	0	2	0	0	0	2	2	2	2
C: (σ’σ)(σ,σ)(,σ)σσσ	0	3!	0	0	0	2	2	2	2

Pass 3

input: (σ’σ)(σ,σ)(σ,σ)σσ	Align-σ-R	Align-σ-L	Align-Max-L	Align-Un-R	NoParse	Align-Ft-L	Align-Ft-R	Align-Min-L	Align-Min-R
☞A: (σ’σ)(σ,σ)(σ,σ)(σ,σ)	0	0	0	0	0	3	3	3	3
B: (σ’σ)(σ,σ)((σ,σ)σ)σ	0	1!	1	0	0	2	2	1	2
C:(σ’σ)(σ,σ)(,σ)(σ,σ)(,σ)σ	0	1!	0	0	0	3	3	3	3
D: ((σ’σ)σ)(σ,σ)(σ,σ)(,σ)	0	0	0	0	0	3	3	3	2

Pass 4

As shown above, the ranking in (99) ends up being a simple binary stress system (with a unary parse for any leftover odd syllable). For comparison, candidate D is the optimal candidate in Parallel OT: it would win out over candidate A by virtue of having one less minimal foot, which means Align-Min-R incurs fewer violations than for a purely binary parse. This candidate is never generated by Serial OT. This is another case of consistent/persistent generation – once an ILT foot is built Serial OT will keep trying to build ILT feet. This means Serial OT could end up with one ILT foot and one binary foot as the process comes to a halt (shown in category I later), but never with one ILT foot and two binary feet (and an additional unary foot).

Finally, Serial OT does also not generate a **T*(B/U)(B)** stress pattern. One constraint ranking leading to such a pattern in Parallel OT is shown in (102).

(102) Align-σ-L, Align-σ-R, Align-Un-R, Align-FtMain-R, MaxTrochee, Parse >> Align-Ft-L, Align-Ft-R, Align-FtMain-L >> Align-Max-L >> Align-Min-L, Align-Min-R >> Align-Max-R, Align-Un-L, MinIamb, MaxIamb >> MinTrochee

Looking at the first stratum, it is visible that maximum parsing is desired because of the presence of both Align-σ-L and Align-σ-R. Since Align-Ft-L/R are in the second stratum and Align-Min-L/R are in the fourth stratum there is little resistance to building additional binary (minimal) feet to maximize parsing. Thus, the constraint ranking in (102) too results in a binary stress system, like the one shown in (101).

There is one more pattern in the E category that needs to be discussed. It is the pattern in (102), generated by PGA and PGAP.

(103)

	T(U)(B*)
	(σσ)
	(σσσ)
	(σσσ)(σ)
	(σσσ)(σσ)
	(σσσ)(σ)(σσ)
	(σσσ)(σσ)(σσ)
	(σσσ)(σ)(σσ)(σσ)
	Unattested
PCA	0
PGAP	16
PGA	16
SCA	0
SGA	0

It is similar to **T(B*)(U)**, except the unary foot is positioned between the ILT foot and the binary foot/feet. A constraint ranking responsible for this pattern is shown in (104).

(104) Align-σ-L, Align-σ-R, Align-Max-L, Align-FtMain-R, MaxTrochee >> Align-Ft-L, Align-Ft-R, Align-Min-L, Align-Min-R Align-FtMain-L >> Align-Max-R, Align-Un-L, MinIamb, MaxIamb >> Align-Un-R, MinTrochee

(σ’σ)	((σ’σ)σ)	((σ,σ)σ)(‘σ)	((σ,σ)σ)(σ’σ)	((σ,σ)σ)(,σ)(σ’σ)	((σ,σ)σ)(σ,σ)(σ’σ)	((σ,σ)σ)(,σ)(σ,σ)(σ’σ)
B	**T**	**TU**	**TB**	**TUB**	**TBB**	**TUBB**

The reason Serial OT does not generate this pattern is consistent/persistent generation: if a five-syllable input results in a TB output (an ILT foot and a binary foot: the largest foot that can be built in one step), then a six-syllable input will never not start by building an ILT foot and a binary foot (and most likely using that binary foot to build an ILT foot).

The table in (105) below shows what PGA, PGAP and PCA make of this constraint ranking. For PGAP, it is assumed that Parse-σ is in its own stratum, below all the other strata. The candidates with two ILT feet are eliminated instantly due to Align-Max-L being in the top stratum. The same is true for unparsed syllables and Align-L/R

(105) Constraint ranking in (104) for PGA, PGAP and PCA, 6 and 7 syllables

Input: σσσσσσ	Align-σ-R	Align-σ-L	Align-Max-L	Align-Ft-L	Align-Ft-R	Align-Min-L	Align-Min-R	Align-Max-R	Align-Un-L	Align-Un-R	Parse-σ
☞PGA1: ((σ,σ)σ)(,σ)(σ’σ)	0	0	0	3	3	3	1	2	1	1	~~0~~
PGA2: ((σ,σ)σ)(σ,σ)(’σ)	0	0	0	3	3	3	1	2	2!	0	~~0~~
PGA3: ((σ,σ)σ)((σ’σ)σ)	0	0	1!	1	1	0	0	1	0	0	~~0~~
☞PGAP1: ((σ,σ)σ)(,σ)(σ’σ)	0	0	0	3	3	3	1	2	1	1	0
PGAP2: ((σ,σ)σ)(σ,σ)(’σ)	0	0	0	3	3	3	1	2	2!	0	0
PGAP3: ((σ,σ)σ)((σ’σ)σ)	0	0	1!	1	1	0	0	1	0	0	0
PCA1: ((σ,σ)σ)(,σ)(σ’σ)	0	0	0	2	2	2	1	1	1	1!	0
☞PCA2: ((σ,σ)σ)(σ,σ)(’σ)	0	0	0	2	2	2	1	1	1	0	0
PCA3: ((σ,σ)σ)((σ’σ)σ)	0	0	1!	1	1	0	0	1	0	0	0

Input: σσσσσσσσ	Align-σ-R	Align-σ-L	Align-Max-L	Align-Ft-L	Align-Ft-R	Align-Min-L	Align-Min-R	Align-Max-R	Align-Un-L	Align-Un-R	Parse-σ
☞PGA1:((σ,σ)σ)(,σ)(σ,σ)(σ’σ)	0	0	0	6	6	6	3	3	1	2	~~0~~
PGA2: ((σ,σ)σ)(σ,σ)(,σ)(σ’σ)	0	0	0	6	6	6	3	3	2!	1	~~0~~
PGA3: ((σ,σ)σ)(σ,σ)(σ,σ)(‘σ)	0	0	0	6	6	6	3	3	3!	0	~~0~~
☞PGAP1:((σ,σ)σ)(,σ)(σ,σ)(σ’σ)	0	0	0	6	6	6	3	3	1	2	0
PGAP2: ((σ,σ)σ)(σ,σ)(,σ)(σ’σ)	0	0	0	6	6	6	3	3	2!	1	0
PGAP3: ((σ,σ)σ)(σ,σ)(σ,σ)(‘σ)	0	0	0	6	6	6	3	3	3!	0	0
PCA1: ((σ,σ)σ)(,σ)(σ,σ)(σ’σ)	0	0	0	3	3	3	2	1	1	1!	0
PCA2: ((σ,σ)σ)(σ,σ)(,σ)(σ’σ)	0	0	0	3	3	3	2	1	1	1!	0
☞PCA3: ((σ,σ)σ)(σ,σ)(σ,σ)(‘σ)	0	0	0	3	3	3	2	1	1	0	0

PCA does not generate the same pattern PGAP and PGA do: instead, it generates a **T(B*)(U)** pattern. The gradient properties of the PGA(P) constraints allow the **T(U)(B*)** to be generated. While for PCA, the candidates are tied until Align-Un-R in the fourth stratum, the tie is broken sooner for PGA(P), however. Since Align-Un-L/R care about the amount of interfering feet, the position of the unary foot is crucial. Align-Un-L outranks Align-Un-R, which means that the unary foot is most optimal next to the ILT foot (since Align-Max-L outranks Align-Un-L). For PCA, the amount of interfering feet does not matter, and since the ILT foot is always to the left of the unary foot it does not matter how many other feet are interfering. The result is that the tie is carried down to Align-Ft-R, where the candidate with the rightmost unary foot is the winner.

F. Mixed ternary/binary systems

(106)	**T*(B)(B)**	**T*(B)(B/T)**	**(B)T*(B)**
	(σσ)	(σσ)	(σσ)
	(σσσ)	(σσσ)	(σσσ)
	(σσ)(σσ)	(σσ)(σσ)	(σσ)(σσ)
	(σσσ)(σσ)	(σσσ)(σσ)	(σσσ)(σσ)
	(σσσ)(σσσ)	(σσσ)(σσσ)	(σσσ)(σσσ)
	(σσσ)(σσ)(σσ)	(σσσ)(σσ)(σσ)	(σσ)(σσσ)(σσ)
	(σσσ)(σσσ)(σσ)	(σσσ)(σσ)(σσσ)	(σσσ)(σσσ)(σσ)
	Estonian, Chugach Alutiiq	Unattested	Unattested
PCA	16	16	16
PGAP	16	0	0
PGA	16	0	0
SCA	0	0	0
SGA	0	0	0

The mixed ternary/binary system presents trouble for Serial OT and PCA. Serial OT is unable to generate any of the patterns in category F, while PCA generates two additional unattested patterns: **T*(B)(B/T)** and **(B)T*(B)**. In (107), (108) and (109) are constraint rankings for PCA for each of the mixed binary/ternary patterns, together with their actual outputs. It must be noted that each output uses the exact same foot structures across the three stress systems and that only the order of these feet is different: the binary foot in the eight-syllable input is the middle foot in **T*(B)(B/T)**, and the ternary foot is in the middle for the seven-syllable input for **(B)T*(B)**.

(107) Align-σ-L, Align-σ-R, Align-Un-L, Align-Un-R, Align-FtMain-R, MinIamb, MaxTrochee, Parse >> Align-Ft-L, Align-Ft-R, **Align-Min-R**, Align-FtMain-L, MinTrochee >> <u>Align-Max-L</u>, **Align-Max-R**, MaxIamb >> <u>Align-Min-L</u>

(σ’σ)	((σ’σ)σ)	(σ,σ)(σ’σ)	((σ,σ)σ)(σ’σ)	((σ,σ)σ)((σ’σ)σ)	((σ,σ)σ)(σ,σ)(σ’σ)	((σ,σ)σ)((σ,σ)σ)(σ’σ)
B	**T**	**BB**	**TB**	**TT**	**TBB**	**TTB**

(108) Align-σ-L, Align-σ-R, Align-Un-L, Align-Un-R, Align-FtMain-R, MinIamb, MaxTrochee, Parse >> Align-Ft-L, Align-Ft-R, Align-FtMain-L, MinTrochee >> <u>Align-Max-L</u>, MaxIamb >> **Align-Max-R**, <u>Align-Min-L</u> >> **Align-Min-R**

(σ’σ)	((σ’σ)σ)	(σ,σ)(σ’σ)	((σ,σ)σ)(σ’σ)	((σ,σ)σ)((σ’σ)σ)	((σ,σ)σ)(σ,σ)(σ’σ)	((σ,σ)σ)(σ,σ)((σ’σ)σ)
B	**T**	**BB**	**TB**	**TT**	**TBB**	**TBT**

(109) Align-σ-L, Align-σ-R, Align-Un-L, Align-Un-R, Align-FtMain-R, MinIamb, MaxTrochee, Parse >> Align-Ft-L, Align-Ft-R, **Align-Min-R**, Align-FtMain-L, MinTrochee >> **Align-Max-R**, <u>Align-Min-L</u>, MaxIamb >> <u>Align-Max-L</u>

(σ’σ)	((σ’σ)σ)	(σ,σ)(σ’σ)	((σ,σ)σ)(σ’σ)	((σ,σ)σ)((σ’σ)σ)	(σ,σ)((σ,σ)σ)(σ’σ)	((σ,σ)σ)((σ,σ)σ)(σ’σ)
B	**T**	**BB**	**TB**	**TT**	**BTB**	**TTB**

The first stratum is identical across all three rankings. The only constraints that vary across rankings are Align-Min-L/R and Align-Max-L/R. The only difference between (107) and (109) is a straight swap between Align-Max-L and Align-Min-L (underlined). The ranking in (108) differs from the other two on more accounts. Most notably, it has five strata, and the last one consists of only Align-Min-R, which is higher in the other two rankings – in the second stratum in fact.

The tableaux in (110) show what output the ranking in (108) produces in PGAP for an eight-syllable input. For comparison, PCA is shown below also.

(110) Constraint ranking in (108) for PGA and PCA, 8 syllables

Input: σσσσσσσσ	Align-σ-L	Align-σ-R	Align-Un-L	Align-Un-R	Parse	Align-Ft-L	Align-Ft-R	Align-Max-L	Align-Max-R	Align-Min-L
☞PGA1:((σ,σ)σ)((σ,σ)σ)(σ’σ)	0	0	0	0	0	3	3	1	3	2
PGA2: (σ,σ)((σ,σ)σ)((σ’σ)σ)	0	0	0	0	0	3	3	3!	1	0
PGA3: ((σ,σ)σ)(σ,σ)((σ’σ)σ)	0	0	0	0	0	3	3	2!	2	1
PCA1: ((σ,σ)σ)((σ,σ)σ)(σ’σ)	0	0	0	0	0	2	2	1	2	1!
PCA2: (σ,σ)((σ,σ)σ)((σ’σ)σ)	0	0	0	0	0	2	2	2!	1	0
☞PCA3:((σ,σ)σ)(σ,σ)((σ’σ)σ)	0	0	0	0	0	2	2	1	1	1

What happens when the constraint ranking in (108) encounters a seven-syllable input in PGAP and PCA is shown in (111).

(111) Constraint ranking in (102) for PGAP and PCA, 7 syllables

Input: σσσσσσσ	Align-σ-L	Align-σ-R	Align-Un-L	Align-Un-R	Parse	Align-Ft-L	Align-Min-R	Align-Ft-R	Align-Max-R	Align-Min-L	Align-Max-L
☞PGA1:((σ,σ)σ)(σ,σ)(σ’σ)	0	0	0	0	0	3	1	3	2	3	0
PGA2: (σ,σ)((σ,σ)σ)(σ’σ)	0	0	0	0	0	3	2	3!	1	2	1
PGA3: (σ,σ)(σ,σ)((σ’σ)σ)	0	0	0	0	0	3	3	3!	0	1	2
PCA1: ((σ,σ)σ)(σ,σ)(σ’σ)	0	0	0	0	0	2	1	2	1	2!	0
☞PCA2:(σ,σ)((σ,σ)σ)(σ’σ)	0	0	0	0	0	2	1	2	1	1	1
PCA3: (σ,σ)(σ,σ)((σ’σ)σ)	0	0	0	0	0	2	2	2!	0	1	1

(110) and (111) both show that the directional irregularities in **T*(B)(B/T)** and **(B)T*(B)**, compared to **T*(B)(B)** disappear and in fact produce forms as in **T*(B)(B)**.

Because both Align-σ-L/R constraints and parse are in the first stratum, and no constraints that incur violations for minimal feet are, for Serial OT, the constraint rankings in (107), (108) and (109) again produce stress system consisting of binary feet and an ILT foot (both Align-Un-L/R constraints are in the first stratum) to parse any remaining odd syllable. This is shown in (112) below for the **T*(B)(B)** pattern in (107).

(112) Constraint ranking in (107) for Serial OT, 7 syllables

Input: σσσσσσσ	Align-σ-L	Align-σ-R	Align-Un-L	Align-Un-R	NoParse	Align-Ft-L	Align-Min-R	Align-Ft-R	Align-Max-R	Align-Max-L	Align-Min-L
A: σσσσσσσ	0	0	0	0	10!	0	0	0	0	0	0
☞B: σσσσσ(σ’σ)	0	5	0	0	0	0	0	0	0	0	0
☞C: (σ’σ)σσσσσ	5	0	0	0	0	0	0	0	0	0	0

Pass 1

Input: (σ’σ)σσσσσ, σσσσσ(σ’σ)	Align-σ-L	Align-σ-R	Align-Un-L	Align-Un-R	NoParse	Align-Ft-L	Align-Min-R	Align-Ft-R	Align-Max-R	Align-Max-L	Align-Min-L
☞A:σσσ(σ,σ)(σ’σ)	0	3	0	0	0	1	1	1	0	0	1
B: σσσσσ(σ’σ)	0	5!	0	0	0	0	0	0	0	0	0
C: (σ’σ)σσσσσ	5!	0	0	0	0	0	0	0	0	0	0
☞D:(σ,σ)(σ’σ)σσσ	3	0	0	0	0	1	1	1	0	0	1
E: ((σ’σ)σ)σσσσ	4!	0	0	0	0	0	0	0	0	0	0

Pass 2

Input: σσσ(σ,σ)(σ’σ), (σ,σ)(σ’σ)σσσ	Align-σ-L	Align-σ-R	Align-Un-L	Align-Un-R	NoParse	Align-Ft-L	Align-Min-R	Align-Ft-R	Align-Max-R	Align-Max-L	Align-Min-L
A: σσσ(σ,σ)(σ’σ)	0	3!	0	0	0	1	1	1	0	0	1
B: (σ,σ)(σ’σ)σσσ	3!	0	0	0	0	1	1	1	0	0	1
☞C:(σ,σ)(σ,σ)(σ’σ)σ	1	0	0	0	0	2	2	2	0	0	2
☞D:σ(σ,σ)(σ,σ)(σ’σ)	0	1	0	0	0	2	2	2	0	0	2

Pass 3

Input: σ(σ,σ)(σ,σ)(σ’σ), (σ,σ)(σ,σ)(σ’σ)σ	Align-σ-L	Align-σ-R	Align-Un-L	Align-Un-R	Max Trochee	Align-Ft-L	Align-Min-R	Align-Ft-R	Align-Max-R	Align-Max-L	Align-Min-L
A:(σ,σ)(σ,σ)(σ’σ)σ	1!	0	0	0	0	2	2	2	0	0	2
B:σ(σ,σ)(σ,σ)(σ’σ)	0	1!	0	0	0	2	2	2	0	0	2
☞C:(σ,σ)(σ,σ)((σ’σ)σ)	0	0	0	0	0	2	2	2	0	0	1
D: (σ(σ,σ))(σ,σ)(σ’σ)	0	0	0	0	1!	2	1	2	0	0	2

Pass 4

Convergence happens after pass 4, the outcome being **BBT.** Note that in the final tableau, NoParse is left out and instead MaxTrochee is shown to account for the dismissal of candidate D, which has an ILT foot with the dependent preceding the minimal foot.

G. Ternary & Binary reparse systems

(113)	**T*(B/σ)**	**(B)T*(σ)**
	(σσ)	(σσ)
	(σσσ)	(σσσ)
	(σσσ)σ	(σσσ)σ
	(σσσ)(σσ)	(σσ)(σσσ)
	(σσσ)(σσσ)	(σσσ)(σσσ)
	(σσσ)(σσσ)σ	(σσσ)(σσσ)σ
	(σσσ)(σσσ)(σσ)	(σσ)(σσσ)(σσσ)
	Tripura Bangla, Winnebago	Unattested
PCA	16	16
PGAP	16	16
PGA	16	16
SCA	8	0
SGA	8	0

A non-directionality issue similar to the one occurring in **(B)T*(U)** in category E occurs here for **(B)T*(σ)**. However, this time PGA(P) can generate this pattern whereas Serial OT cannot.

(114) Align-σ-R, **Align-Min-L**, Align-Un-L, Align-Un-R, Align-FtMain-L, MinTrochee, MaxTrochee >> Stray-Ft-L, Parse >> Align-Ft-L, Align-Ft-R, Align-Max-L, Align-Max-R, **Align-Min-R**, Align-FtMain-R, MinIamb, MaxIamb

(‘σσ)	((‘σσ)σ)	((‘σσ)σ)σ	(‘σσ)((,σσ)σ)	((‘σσ)σ)((,σσ)σ)	((‘σσ)σ)((,σσ)σ)σ	(‘σσ)((,σσ)σ)((,σσ)σ)
B	**T**	**Tσ**	**BT**	**TT**	**TTσ**	**BTT**

Align-Min-L outranks Align-Min-R. The two constraints are swapped for a **T*(B/σ)** output pattern. This is reflected in the actual output patterns: with Align-Min-L in the top stratum minimal feet are encouraged to appear in initial position, while Align-Min-R puts them in final position. A similar effect is observed should Align-σ-R and Align-σ-L change positions.

The tableaux in (115) below show what happens when this constraint ranking is encountered with Serial OT. Due to Align-Un-L/R both being in the first stratum, any forms with unary feet are not considered here.

(115)

Input: σσσσσσσ	Align-σ-R	Align-Min-L	Align-Un-L	Align-Un-R	Align-σ-L	NoParse	Align-Ft-L	Align-Ft-R	Align-Max-R	Align-Max-L	Align-Min-R
σσσσσσσ	0	0	0	0	0	10!	0	0	0	0	0
σσσσσ(ʻσσ)	5!	0	0	0	0	5	0	0	0	0	0
☞(ʻσσ)σσσσσ	0	0	0	0	5	0	0	0	0	0	0
Input: (ʻσσ)σσσσσ	Align-σ-R	Align-Min-L	Align-Un-L	Align-Un-R	Align-σ-L	NoParse	Align-Ft-L	Align-Ft-R	Align-Max-R	Align-Max-L	Align-Min-R
(ʻσσ)σσσσσ	0	0	0	0	5!	0	0	0	0	0	0
(ʻσσ)(ˌσσ)σσσ	0	1!	0	0	3	0	1	1	0	0	1
☞((ʻσσ)σ)σσσσ	0	0	0	0	4	0	0	0	0	0	0
Input: ((ʻσσ)σ)σσσσ	Align-σ-R	Align-Min-L	Align-Un-L	Align-Un-R	Align-σ-L	NoParse	Align-Ft-L	Align-Ft-R	Align-Max-R	Align-Max-L	Align-Min-R
((ʻσσ)σ)(ˌσσ)σσ	0	1!	0	0	2	0	1	1	1	0	0
☞ ((ʻσσ)σ)σσσσ	0	0	0	0	4	0	0	0	0	0	0

A curious interaction between Align-σ-R and Align-Min-L causes the halt of any parsing beyond an initial ILT foot. Any foot that is created after that is placed to the right of the initial foot, not violating Align-σ-R, but incurring marks for Align-Min-L. Swapping the Align-Min-R and Align-Min-L constraints allows the binary foot in the third pass to be built, and later expanded upon again, leading to a **T*(B/σ)** pattern.

H. Strictly ternary system

(116)	**T*(σ)**
	(σσ)
	(σσσ)
	(σσσ)σ
	(σσσ)σσ
	(σσσ)(σσσ)
	(σσσ)(σσσ)σ
	(σσσ)(σσσ)σσ
	Cayuvava, Gilbertese, Sentani
PCA	16
PGAP	16
PGA	16
SCA	0
SGA	0

Serial OT is unable to generate ternary-only rhythm. The ranking for one such system for Parallel OT is shown in (117).

(117) Align-σ-R, Align-Min-L, Align-Min-R, Align-Un-L, Align-Un-R, Align-Un-L, Align-FtMain-L, MaxIamb, MaxTrochee >> Align-σ-L, Parse >> Align-Ft-L, Align-Ft-R, Align-Max-L, Align-Max-R, Align-FtMain-R, MinTrochee, MaxIamb

(σ'σ)	((σ'σ)σ)	((σ'σ)σ)σ	((σ'σ)σ)σσ	((σ'σ)σ)((σ,σ)σ)	((σ'σ)σ)((σ,σ)σ)σ	((σ'σ)σ)((σ,σ)σ)σσ
B	**T**	**Tσ**	**Tσσ**	**TT**	**TTσ**	**TTσσ**

For Serial OT with NoParse, the following happens.

(118) Constraint ranking in (117) for Serial OT

Input: σσσσσσ	Align-σ-R	Align-Min-L	Align-Min-R	Align-Un-R	Align-Un-L	NoParse	Align-σ-L	Align-Ft-R	Align-Max-R	Align-Max-L	Align-Ft-L
☞(σ'σ)σσσσ	0	0	0	0	0	0	4	0	0	0	0
σσσσσσ	0	0	0	0	0	10!	0	0	0	0	0

Pass 1

Input: (σ'σ)σσσσ	Align-σ-R	Align-Min-L	Align-Min-R	Align-Un-R	Align-Un-L	NoParse	Align-σ-L	Align-Ft-R	Align-Max-R	Align-Max-L	Align-Ft-L
(σ'σ)σσσσ	0	0	0	0	0	0	4!	0	0	0	0
(σ'σ)(σ,σ)σσ	0	1!	1	0	0	0	2	1	0	0	1
☞((σ'σ)σ)σσσ	0	0	0	0	0	0	3	0	0	0	0

Pass 2

Input: ((σ'σ)σ)σσσ	Align-σ-R	Align-Min-L	Align-Min-R	Align-Un-R	Align-Un-L	Parse	Align-σ-L	Align-Ft-R	Align-Max-R	Align-Max-L	Align-Ft-L
☞((σ'σ)σ)σσσ	0	0	0	0	0	0	3	0	0	0	0
((σ'σ)σ)(σ,σ)σ	0	1!	0	0	0	0	1	1	1	0	1

Pass 3

For the same reason as in the G category, Serial OT only generates one maximal foot and then stops creating additional feet. Adding another foot would violate at least one top-ranked constraint, which is more than the zero constraints the status quo at that point violates. A second maximal foot would be a harmonic improvement over a single one since it would mean Align-σ-L in the second stratum receives fewer violations, but alas, Serial OT cannot generate an ILT foot in one step. Moving Align-Min-L to the second stratum does allow the binary foot to be built after the initial ILT foot. However, due to the local nature of Serial OT, this would also happen in a five-syllable input, resulting in an output pattern of **T*(B/σ)**.

I. Mixed ternary/binary triple stray systems

(119)	**T(B)(σ*)**	**T*(B/U)(σ*)**	**(B)T(σ*)**	**(B/U)T(σ*)**
	(σσ)	(σσ)	(σσ)	(σσ)
	(σσσ)	(σσσ)	(σσσ)	(σσσ)
	(σσσ)σ	(σσσ)(σ)	(σσσ)σ	(σ)(σσσ)
	(σσσ)(σσ)	(σσσ)(σσ)	(σσ)(σσσ)	(σσ)(σσσ)
	(σσσ)(σσ)σ	(σσσ)(σσ)σ	(σσ)(σσσ)σ	(σσ)(σσσ)σ
	(σσσ)(σσ)σσ	(σσσ)(σσ)σσ	(σσ)(σσσ)σσ	(σσ)(σσσ)σσ
	(σσσ)(σσ)σσσ	(σσσ)(σσ)σσσ	(σσ)(σσσ)σσσ	(σσ)(σσσ)σσσ
	Unattested	Unattested	Unattested	Unattested
PCA	16	16	16	16
PGAP	16	16	16	16
PGA	0	0	0	0
SCA	8	8	0	0
SGA	8	8	0	0

Here is visible the first difference between PGA and PGAP. While PGAP can match any pattern generated by PCA in this category, PGAP cannot generate a single such pattern. Meanwhile, Serial OT generates eight **T(B)(σ*)** patterns, and eight **T*(B/U)(σ*)**, but not any of the remaining two patterns.

First, Serial OT will be looked at. The patterns it cannot generate have, again, a non-directionality component to them. For **(B)T(σ*)** and **(B/U)T(σ*)**, the extra binary/unary foot is at the other side of the IL than the unparsed syllables, while for **T(B)(σ*)** and **T*(B/U)(σ*)** they are on the same side. In (120), (121), (122) and (123) are rankings for the above patterns.

(120) Align-σ-R, Align-Max-L, Align-Min-R, Align-Un-L, Align-Un-R, Align-FtMain-L, MinTroch, MaxTrochee>>Align-σ-L, Parse >> Align-Ft-L, Align-Ft-R, Align-Max-R, Align-Min-L, Align-FtMain-R, MinIamb, MaxIamb

(‘σσ)	((‘σσ)σ)	((‘σσ)σ)σ	((‘σσ)σ)(,σσ)	((‘σσ)σ)(,σσ)σ	((‘σσ)σ)(,σσ)σσ	((‘σσ)σ)(,σσ)σσσ
B	**T**	**Tσ**	**TB**	**TBσ**	**TBσσ**	**TBσσσ**

(121) Align-σ-R, Align-Max-L, Align-Min-R, Align-Un-R, Align-FtMain-L, MinTroch, MaxTrochee>>Align-σ-L, Parse >> Align-Ft-L, Align-Ft-R, Align-Max-R, Align-Min-L, Align-Un-L, Align-FtMain-R, MinIamb, MaxIamb

(‘σσ)	((‘σσ)σ)	((‘σσ)σ)(,σ)	((‘σσ)σ)(,σσ)	((‘σσ)σ)(,σσ)σ	((‘σσ)σ)(,σσ)σσ	((‘σσ)σ)(,σσ)σσσ
B	**T**	**TU**	**TB**	**TBσ**	**TBσσ**	**TBσσσ**

(122) Align-σ-R, Align-Max-R, Align-Min-L, Align-Un-L, Align-Un-R, Align-FtMain-L, MinTroch, MaxTrochee>>Align-σ-L, Parse >> Align-Ft-L, Align-Ft-R, Align-Max-L, Align-Min-R, Align-FtMain-R, MinIamb, MaxIamb

(‘σσ)	((‘σσ)σ)	((‘σσ)σ)σ	(‘σσ)((,σσ)σ)	(‘σσ)((,σσ)σ)σ	(‘σσ)((,σσ)σ)σσ	(‘σσ)((,σσ)σ)σσσ
B	**T**	**Tσ**	**BT**	**BTσ**	**BTσσ**	**BTσσσ**

(123) Align-σ-R, Align-Max-R, Align-Min-L, Align-Un-L, Align-FtMain-L, MinTroch, MaxTrochee>>Align-σ-L, Parse >> Align-Ft-L, Align-Ft-R, Align-Max-L, Align-Min-R, Align-Un-R, Align-FtMain-R, MinIamb, MaxIamb

(‘σσ)	((‘σσ)σ)	(‘σ)((,σσ)σ)	(‘σσ)((,σσ)σ)	(‘σσ)((,σσ)σ)σ	(‘σσ)((,σσ)σ)σσ	(‘σσ)((,σσ)σ)σσσ
B	**T**	**UT**	**BT**	**BTσ**	**BTσσ**	**BTσσσ**

The tableaux (124) below shows what happens when the ranking in (123) is used for Serial OT.

(124) Constraint ranking in (123) for Serial OT, 4 syllables

Input: σσσσ	Align-σ-R	Align-Max-R	Align-Min-L	Align-Un-L	Align-σ-L	NoParse	Align-Ft-L	Align-Ft-R	Align-Max-L	Align-Min-R	Align-Un-R
☞(‘σσ)σσ	0	0	0	0	2	0	0	0	0	0	0
σσσσ	0	0	0	0	0	10!	0	0	0	0	0
(‘σ)σσσ	0	0	0	0	3!	0	0	0	0	0	0
σ(‘σσ)σ	1!	0	0	0	1	0	0	0	0	0	0

Pass 1

Input: (‘σσ)σσ	Align-σ-R	Align-Max-R	Align-Min-L	Align-Un-L	Align-σ-L	NoParse	Align-Ft-L	Align-Ft-R	Align-Max-L	Align-Min-R	Align-Un-R
☞((‘σσ)σ)σ	0	0	0	0	1	0	0	0	0	0	0
(‘σσ)(,σσ)	0	0	1!	0	0	0	1	1	0	1	0
(‘σσ)σσ	0	0	0	0	2!	0	0	0	0	0	0

Pass 2

Input: ((‘σσ)σ)σ	Align-σ-R	Align-Max-R	Align-Min-L	Align-Un-L	Align-σ-L	NoParse	Align-Ft-L	Align-Ft-R	Align-Max-L	Align-Min-R	Align-Un-R
☞((‘σσ)σ)σ	0	0	0	0	1	0	0	0	0	0	0
((‘σσ)σ)(,σ)	0	1!	1	1	0	0	1	1	0	0	0

Pass 3

After the first pass it is already impossible to generate the same pattern as in (123), due to the binary foot not being in the middle – again a case of edge-undergeneration. A candidate with an initial unary foot is also turned down due to too many violations for Align-σ-L. The end result, for a four-syllable input, consists of an initial ILT foot and an unparsed syllable (the same output as the ranking in (122) would have had, where Align-Un-R is also in the first stratum). For more syllables, the initial ILT foot would remain, and no additional feet are crafted. This is illustrated in (125).

(125) Constraint ranking in (122) for Serial OT, 8 syllables

Input: σσσσσσσσ	Align-σ-R	Align-Max-R	Align-Min-L	Align-Un-L	Align-σ-L	NoParse	Align-Ft-L	Align-Ft-R	Align-Max-L	Align-Min-R	Align-Un-R
☞(ˈσσ)σσσσσσ	0	0	0	0	6	0	0	0	0	0	0
σσσσσσσσ	0	0	0	0	0	10!	0	0	0	0	0
(ˈσ)σσσσσσσ	0	0	0	0	7!	0	0	0	0	0	0
σ(ˈσσ)σσσσσ	1!	0	0	0	6	0	0	0	0	0	0

Pass 1

Input: (ˈσσ)σσσσσσ	Align-σ-R	Align-Max-R	Align-Min-L	Align-Un-L	Align-σ-L	NoParse	Align-Ft-L	Align-Ft-R	Align-Max-L	Align-Min-R	Align-Un-R
☞((ˈσσ)σ)σσσσσ	0	0	0	0	5	0	0	0	0	0	0
(ˈσσ)(ˌσσ)σσσσ	0	0	1!	0	0	0	1	1	0	1	0
(ˈσσ)σσσσσσ	0	0	0	0	6!	0	0	0	0	0	0

Pass 2

Input: ((ˈσσ)σ)σσσσσ	Align-σ-R	Align-Max-R	Align-Min-L	Align-Un-L	Align-σ-L	NoParse	Align-Ft-L	Align-Ft-R	Align-Max-L	Align-Min-R	Align-Un-R
☞((ˈσσ)σ)σσσσσ	0	0	0	0	5	0	0	0	0	0	0
((ˈσσ)σ)(ˌσ)σσσσ	0	1!	1	1	4	0	1	1	0	0	0
((ˈσσ)σ)(ˌσσ)σσσ	0	1!	1	0	3	0	1	1	0	0	0

Pass 3

Again, the same output would have emerged had Align-Un-R been in the first stratum.

The difference between PGA and PGAP is shown in (126), with the ranking as in (121).

(126) Constraint ranking in (121) for PGAP and PGA, 7 syllables

Input: σσσσσσσ	Align-σ-R	Align-Max-L	Align-Min-R	Align-Un-R	Align-σ-L	Parse	Align-Ft-L	Align-Ft-R	Align-Max-R	Align-Min-L	Align-Un-L
☞PGAP1:((ˈσσ)σ)(ˌσσ)σσ	0	0	0	0	4	2	1	1	1	1	0
PGAP2:((ˈσσ)σ)((ˌσσ)σ)σ	0	1!	0	0	2	1	1	1	1	0	0
PGAP3:((ˈσσ)σ)(ˌσσ)(ˌσσ)	0	0	1!	0	0	0	3	3	2	3	0
PGAP4:((ˈσσ)σ)σσσσ	0	0	0	0	4	4!	0	0	0	0	0
PGA1:((ˈσσ)σ)(ˌσσ)σσ	0	0	0	0	4	~~2~~	1!	1	1	1	0
PGA2:((ˈσσ)σ)((ˌσσ)σ)σ	0	1!	0	0	1	~~1~~	1	1	1	0	0
PGA3:((ˈσσ)σ)(ˌσσ)(ˌσσ)	0	0	1!	0	0	~~0~~	3	3	2	3	0
☞PGA4:((ˈσσ)σ)σσσσ	0	0	0	0	4	~~4~~	0	0	0	0	0

Due to the gradient nature of the constraints for PGA and PGAP (one violation mark per foot interfering between an unparsed syllable and the left edge), the two unparsed syllables in candidate A in receive four violation marks for Align-σ-L (two feet interfere between the left edge and the unparsed syllables) – equally as many as the four unparsed syllables in candidate D. Parse is the tiebreaker in case of PGAP, but for PGA the tie is carried to the next stratum, where it is settled in favour of candidate D. Since it has only one foot, Align-Ft-L/R are not violated, and neither are Align-Max-R and Align-Min-L. For eight syllables, there is no tie, since the five unparsed syllables each incurring one mark result in less marks than three unparsed syllables each receiving two marks. In case of six syllables or fewer, the same pattern as for PCA occurs, as (127) shows.

(127) Constraint ranking in (121) for PGAP and PGA, 6 syllables

Input: σσσσσσ	Align-σ-R	Align-Max-L	Align-Min-R	Align-Un-R	Align-σ-L	Parse	Align-Ft-L	Align-Ft-R	Align-Max-R	Align-Min-L	Align-Un-L
☞PGAP1:((ʻσσ)σ)(ˌσσ)σ	0	0	0	0	2	2	1	1	1	1	0
PGAP2:((ʻσσ)σ)σσσ	0	0	0	0	3	3!	0	0	0	0	0
PGA1:((ʻσσ)σ)(ˌσσ)σ	0	0	0	0	2	2	1	1	1	1	0
☞PGA2:((ʻσσ)σ)σσσ	0	0	0	0	3!	3	0	0	0	0	0

This pattern is an addition to the I Category. It is not only PGA that generates this pattern, however, as it can also be achieved with PGAP.

I. Mixed ternary/binary triple stray systems (continued)

(128)	**T(B)(σ*) II**	**T*(B/U)(σ*) II**	**(B)T(σ*) II**	**(B/U)T(σ*) II**
	(σσ)	(σσ)	(σσ)	(σσ)
	(σσσ)	(σσσ)	(σσσ)	(σσσ)
	(σσσ)σ	(σσσ)(σ)	(σσσ)σ	(σ)(σσσ)
	(σσσ)(σσ)	(σσσ)(σσ)	(σσ)(σσσ)	(σσ)(σσσ)
	(σσσ)(σσ)σ	(σσσ)(σσ)σ	(σσ)(σσσ)σ	(σσ)(σσσ)σ
	(σσσ)σσσσ	(σσσ)σσσσ	(σσσ)σσσσ	(σσσ)σσσσ
	(σσσ)σσσσσ	(σσσ)σσσσσ	(σσσ)σσσσσ	(σσσ)σσσσσ
	Unattested	Unattested	Unattested	Unattested
PCA	0	0	0	0
PGAP	16	16	16	16
PGA	16	16	16	16
SCA	0	0	0	0
SGA	0	0	0	0

These patterns can also be generated with Parse, however. The ranking in (129) is one that generates such a pattern as in **T(B)(σ*)II**.

(129) Align-σ-R, Align-Max-L, Align-Min-R, Align-Un-R, Align-Un-L, Align-FtMain-L, MinIamb, MaxTrochee >> Align-σ-L >> Align-Ft-L, Align-Ft-R, Align-Max-R, Align-Min-L, Align-FtMain-R, MinTrochee, MaxIamb >> Parse-σ

This ranking differs in one aspect from the one in (120): Parse-σ is demoted all the way down to its own stratum at the bottom of the ranking. In this way, the ranking becomes essentially equal to a PGA ranking, since no candidates are tied until Parse-σ gets involved.

Curiously, an additional pattern emerges. For this pattern, only the eight-syllable output is different from the initial patterns, and only different in the seven-syllable output compared to the previous patterns. These patterns are generated exclusively by PGAP.

I. Mixed ternary/binary triple stray systems (continued (continued))

(130)	**T(B)(σ*) III**	**T*(B/U)(σ*) III**	**(B)T(σ*) III**	**(B/U)T(σ*) III**
	(σσ)	(σσ)	(σσ)	(σσ)
	(σσσ)	(σσσ)	(σσσ)	(σσσ)
	(σσσ)σ	(σσσ)(σ)	(σσσ)σ	(σ)(σσσ)
	(σσσ)(σσ)	(σσσ)(σσ)	(σσ)(σσσ)	(σσ)(σσσ)
	(σσσ)(σσ)σ	(σσσ)(σσ)σ	(σσ)(σσσ)σ	(σσ)(σσσ)σ
	(σσσ)(σσ)σσ	(σσσ)(σσ)σσ	(σσ)(σσσ)σσ	(σσ)(σσσ)σσ
	(σσσ)σσσσσ	(σσσ)σσσσσ	(σσσ)σσσσσ	(σσσ)σσσσσ
	Unattested	Unattested	Unattested	Unattested
PCA	0	0	0	0
PGAP	16	16	16	16
PGA	0	0	0	0
SCA	0	0	0	0
SGA	0	0	0	0

The constraint ranking for **T(B)(σ*)III** in PGAP is shown in (131).

(131) Align-σ-R, Align-Max-L, Align-Min-R, Align-Un-L, Align-Un-R, Align-FtMain-L, MinTrochee, MaxTrochee >> Align-σ-L >> MaxIamb, Parse >> Align-Ft-L, Align-Ft-R, Align-Max-R, Align-Min-L, Align-FtMain-R, MinIamb

(‘σσ)	((‘σσ)σ)	((‘σσ)σ)σ	((‘σσ)σ)(,σσ)	((‘σσ)σ)(,σσ)σ	((‘σσ)σ)(,σσ)σσ	((‘σσ)σ)σσσσσ
B	**T**	**Tσ**	**TB**	**TBσ**	**TBσσ**	**Tσσσσσ**

The tableau in (132) shows what happens for a seven- and eight-syllable input for PGAP, PGA and PCA respectively.

(132) Constraint ranking in (131) for PGAP, PGA and PCA, 7 and 8 syllables

Input: σσσσσσσ	Align-σ-R	Align-Max-L	Align-Min-R	Align-Un-L/R	Align-σ-L	Parse	Align-Ft-L	Align-Ft-R	Align-Max-R	Align-Min-L
☞PGAP1:((‘σσ)σ)(,σσ)σσ	0	0	0	0	4	2	1	1	1	1
PGAP2:((‘σσ)σ)σσσσ	0	0	0	0	4	4!	0	0	0	0
PGA1:((‘σσ)σ)(,σσ)σσ	0	0	0	0	4	$\cancel{2}$	1!	1	1	1
☞PGA2:((‘σσ)σ)σσσσ	0	0	0	0	4	$\cancel{4}$	0	0	0	0
☞PCA1:((‘σσ)σ)(,σσ)σσ	0	0	0	0	2	2	1	1	1	1
PCA2:((‘σσ)σ)σσσσ	0	0	0	0	4!	4	0	0	0	0

Input: σσσσσσσσ	Align-σ-R	Align-Max-L	Align-Min-R	Align-Un-L/R	Align-σ-L	Parse	Align-Ft-L	Align-Ft-R	Align-Max-R	Align-Min-L
PGAP1:((‘σσ)σ)(,σσ)σσσ	0	0	0	0	6!	3	1	1	1	1
☞PGAP2:((‘σσ)σ)σσσσσ	0	0	0	0	5	5	0	0	0	0
PGA1:((‘σσ)σ)(,σσ)σσσ	0	0	0	0	6!	$\cancel{3}$	1	1	1	1
☞PGA2:((‘σσ)σ)σσσσσ	0	0	0	0	5	$\cancel{5}$	0	0	0	0
☞PCA1:((‘σσ)σ)(,σσ)σσσ	0	0	0	0	3	3	1	1	1	1
PCA2:((‘σσ)σ)σσσσσ	0	0	0	0	5!	5	0	0	0	0

PCA and PGA generate different patterns: PGA only allows for one syllable in both seven- and eight-syllable inputs, while PCA generates two syllables in both cases. The reason is that there are two feet intervening in between the unparsed syllables and the left edge for PGA1, PGAP1 and PCA1. Since the gradient constraint Align-σ-L count the number of intervening feet and then assign a violation mark for each of them the amount of violation marks is doubled for PGA compared to PCA, leading to a different output pattern.

J. Unparsed

Serial OT generates an entirely unparsed system. These are not present in Parallel OT because the entirely unparsed forms are not part of the input file. Additionally, Serial OT generates two patterns with all unparsed output except for the two-syllable form, where a binary foot is constructed (one pattern has a trochee, the other an iamb). The constraint ranking for this pattern is shown in (133).

(133) Unparsed except for 2 syllables

> Align-Ft-L, Align-Ft-R, Align-Min-L, Align-Min-R, Align-Max-L, Align-Max-R, Align-Un-L, Align-Un-R, Align-σ-L, Align-σ-R, Align-FtMain-L, Align-FtMain-R, Trochee, NonMinTrochee, NonMinIamb >> Parse-σ >> Iamb

All the constraints except are together in the first stratum, save for Parse-σ and Iamb, who both occupy their own strata. The only violation mark an unparsed form receives is for Parse-σ, equal to the amount of syllables. Violation marks are possibly received for Align-σ-L/R if a foot is constructed and any syllables are left over. This is not the case for a binary foot in a two-syllable input as all syllables are parsed. For any larger inputs the binary foot does not cover every syllable, leading to violation marks for Align-σ-L/R which is ranked higher than Parse-σ and ultimately leaves the entire input unparsed.

I will not remark on Parallel OT in this section, as no unparsed forms were included in the OTSoft input file, which means no unparsed forms were able to surface.

3.3.1 Summary

Table (134) below shows the number of output patterns each method produces for each rhythmic category.

(134)

	PCA	PGAP	PGA	SCA	SGA
A	12	12	12	8	8
B	24	40	40	16	16
C	24	24	24	16	16
D	32	16	16	8	8
E	64	80	80	8	8
F	48	16	16	0	0
G	32	32	32	8	8
H	16	16	16	0	0
I	64	64	192	16	16
J	0	0	0	3	3
Total	316	428	300	82	82

Serial OT produces the fewest number of output patterns. PGA produces more, but still less than PCA and PGAP. With the exception of category J, the output patterns of Serial OT are a proper subset of the output patterns of PCA. By showing the results in section 3.3 I have tried to show why Serial OT is more restrictive: mostly because of edge-undergeneration and consistent/persistent generation. I will discuss this further in chapter 4.

3.4 Duplicates

The discerning reader may have picked up on the need for this section. It is the case that some stress patterns can be described equally well by two (or more) different feet transcriptions. For instance, a word with initial stress can be described by an initial binary foot, and initial ILT foot or even an initial unary foot (though the latter is not generated). A sequence like 01020200 (1 being a syllable with main stress, 2 a syllable with secondary stress and 0 an unstressed syllable) might be the result of any of parses in (135).

(135) Possible parsing of a 01020200 stress pattern

(01)(02)(02)00	(010)(20)(20)0	(010)(20)(200)
0(1)0(2)0(2)00	0(10)(20)(200)	(01)(020)(20)0
(01)(02)(020)0	0(10)(20)(20)0	(01)(020)(200)

This section will explore how many unique output patterns are generated by each of PGA, PGAP, PCA and Serial OT. The numbers are in (136). The header “Unique patterns” refers to the number of patterns this method generates that are unique stress-wise when all footing (i.e. parentheses) is stripped away. If the example in (135) was a real typology, then the amount of unique patterns would be 1: 01020200. The number of duplicates would be 8, since the all patterns are identical to 01020200.

(136) Table showing duplicates

Method	**Patterns**	**Unique patterns**	**Duplicates**
Parallel CA (PCA)	316	288	28
Parallel CA with Parse (PCAP)	316	288	28
Parallel GA with Parse (PGAP)	428	392	36
Parallel GA (PGA)	300	264	36
Serial CA (SCA)	82	63	19
Serial GA (SGA)	82	63	19

Nearly a quarter of output patterns in Serial OT are duplicates, resulting in a mere 63 unique stress patterns. The other methods do not generate 25% of duplicates, although the absolute number does go up. PCA generates 28 duplicates out of 316 patterns and PGA(P) generates 36 duplicates out of 428 and 300 patterns.

3.5 Summary

In summary, Serial OT is the most restrictive method of generating a stress typology, with only 83 possibly languages, of which 63 have unique stress patterns. For Serial OT it does not matter whether GA or CA constraints are used.

For Parallel OT, there does exist a difference between gradient constraints and categorical constraints, and the presence of Parse-σ makes a difference for Parallel OT with GA constraints only.

In chapter 4, I will discuss the results shown in chapter 3. The discussion will revolve around the issue of undergeneration and overgeneration.

Chapter 4. Discussion

The goal of this thesis is to draw comparisons between Parallel OT and Serial OT, and to compare two different types of constraints: gradient and categorical. The table from section 3.6 including duplicate numbers is replicated below in (137).

(137) Table showing duplicates

Method	Patterns	Unique patterns	Duplicates
Parallel CA (PCA)	316	288	28
Parallel CA with Parse (PCAP)	316	288	28
Parallel GA with Parse (PGAP)	428	392	36
Parallel GA (PGA)	300	264	36
Serial CA (SCA)	83	63	20
Serial GA (SGA)	83	63	20

Before continuing, it is important to select the methods to compare. First off, since SCA and SGA yield identical results, these two will be grouped together and left out of the categorical versus gradient discussion. Secondly, and similarly, PCA and PCAP also yield identical results – only PCA will be discussed. Thirdly, it is important to look at Category I to avoid discussion about PGAP, which only differs from PGA in category I. I will do this immediately in the following section.

4.1 Category I

Category I, with mixed ternary/binary triple stray systems, is a large category, with only unattested stress systems. In total, Serial OT generates 16 patterns in this category, PCA and PGA generate 64, and PGAP generates 192 patterns. The table in (138) shows which percentage of all patterns occurs in category I.

(138) Table showing Category I patterns

Method	Patterns	Unique patterns	Category I	% patterns
Parallel CA (PCA)	316	288	64	20
Parallel GA with Parse (PGAP)	428	392	192	45
Parallel GA (PGA)	300	264	64	21
Serial OT (SOT)	83	63	16	19

Serial OT has the fewest patterns in Category I, and it also has the lowest percentage of all patterns occurring in Category I. PCA and PGA have the same number of Category I patterns, while PGAP has triple that number, with a staggering 45% of all PGAP patterns residing in Category I. It must also be noted that the only difference between PGA and PGAP occurs in category I: PGA generates 64 patterns, while PGAP generates 192. The difference of 128 is also the difference between 428 and 300. Indeed, all 300 patterns constructed by PGA are also constructed by PGP. It can be said with certainty that, with all other factors identical, the method resulting in fewer unattested patterns is preferable. Therefore, it makes sense to leave PGAP out of the remainder of the discussion.

The discussion will thus revolve around comparing only PCA, PGA and Serial OT (SOT). In order to be able to say whether any method is preferable, we need to look at what kind of under- and overgeneration they are responsible for. Undergeneration can be classified as the failure to generate patterns that do occur in natural languages, for example the failure to generate strict ternary rhythm. Both undergeneration and overgeneration are undesirable results. Undergeneration is undesirable because it leaves out patterns that do occur in natural languages, and thus leaves typological gaps. When stress systems are generated that do systematically not occur in natural languages that can be called overgeneration. An example of this would be a stress pattern with main stress on the fourth syllable, falling outside the stress window (Kager 2012), or non-directional patterns. Overgeneration is undesirable, like undergeneration, but not as severe. Overgeneration of patterns does not lead to exclusion of naturally occurring languages. An overgeneration can be as innocuous as an unattested stress pattern which is a mirror of an attested stress pattern. The kind of overgeneration that is unwanted is the pathological overgeneration. In chapter 2, the midpoint pathology and the odd-parity input problem have been discussed – both examples of pathological overgeneration, since phenomena like these do not occur in natural languages. Bane & Riggle (2009) put it as “minimizing […] the prediction of ‘unnatural’ or implausible patterns whose lack of attestation is not deemed an accident”. Stanton (2014) calls pathological overgeneration “a problematic prediction of the theory”. Martínez-Paricio & Kager (to appear) use the following definition: “any pattern in which the metrical structure is inconsistent between forms of length *n* versus *n+i* syllables.” A pathological overgeneration can be classified as an overgeneration that poses a problem to the theory by being unnatural or implausible.

First I will look at undergeneration, which is a problem mostly for Serial OT.

4.2 Undergeneration

In the results shown in (134) it can be seen that Serial OT undergenerates a lot of patterns. Many of these are unattested, but there are also attested patterns that are not generated. Of all the languages listed in chapter 3, Serial OT does not generate Macedonian, Azkoitia Basque (both in category A), Indonesian (category D), Kashaya (category E), Estonian, Chugach Alutiiq (both category E), Winnebago (category G), Cayuvava, Gilbertese and Sentani (Category H).

As pointed out in chapter 3, many of these undergenerations stem from one of two issues Serial OT has: **edge-undergeneration** and **consistent/persistent generation**. Both edge-undergeneration and consistent/persistent generation show the limits of Serial OT.

4.2.1 Edge-undergeneration

Edge-undergeneration means that, in Serial OT, a foot will always be constructed as close to the word- or foot edge as possible. This is an issue because it leads to a failure to produce languages with stress on the third syllable such as Macedonian, Cayuvava or Azkiota Basque. Parallel OT does generate these patterns because Parallel OT can look ahead and find the best parsing for the constraint ranking. Serial OT can only make the best single change for the constraint ranking, and building a foot not adjacent to a word edge is never the best single change. One way to counter edge-undergeneration is to adopt constraints assign violation marks for feet built at a word edge. Non-Finality (Kager 2012) and Non-Initiality (Buckley 1997) are constraints do exactly this. Both can and have been used in different ways in the literature. I will use them here as follows.

(139) Non-Finality: Assign one violation mark if the final syllable of a word is part of some foot.

(140) Non-Initiality: Assign one violation mark if the initial syllable of a word is part of some foot.

The table in (141) shows that these constraints can help Serial OT produce stress patterns with stress on the third syllable by re-analysing the stress patterns with unparsed final or initial syllables. The greyed-out row cannot be generated by Serial OT, while the remaining rows can and are.

(141) Cayuvava, Macedonian, Azkoitia Basque

Cayuvava without Non-Finality	(‘σσσ)(’σσσ)	σ(’σσσ)(’σσσ)	σσ(’σσσ)(’σσσ)
Cayuvava with Non-Finality	(‘σσ)(σ’σσ)σ	(σ’σσ)(σ’σσ)σ	σ(σ’σσ)(σ’σσ)σ
Macedonian without Non-Finality	σσσ(’σσσ)	σσσσ(’σσσ)	σσσσσ(’σσσ)
Macedonian with Non-Finality	σσσ(’σσ)σ	σσσσ(’σσ)σ	σσσσσ(’σσ)σ
Azkoitia Basque without Non-Initiality	(σ(σ’σ))σσσ	(σ(σ’σ))σσσσ	(σ(σ’σ))σσσσ
Azkoitia Basque with Non-Initiality	σ(σ’σ)σσσ	σ(σ’σ)σσσσ	σ(σ’σ)σσσσ

The downside of adding these two constraints is that it can lead to overgeneration. OTHelp shows that adding Non-Finality and Non-Initiality causes Serial OT to generate 1763 stress patterns, 786 of which are unique. Many of these are irregular and unattested. For instance, the combination of Non-Finality and Non-Initiality also leads many two-syllable inputs to remain unparsed. Another example is shown in (142), where, for a four-syllable input a binary foot is generated away from both edges and the left and right unparsed feet are later parsed with unary feet.

(142) NonMinTrochee, NonMinIamb >> Parse-σ >>Non-Finality, Non-Initiality >> Align-σ-L

Input: /σσσσ/	Parse-σ	Non-Finality	Non-Initiality	Align-σ-L
☞A: σ(σσ)σ	**			*
B: (σσ)σσ	**		*!	**
C: σσ(σσ)	**	*!		
D: σσσσ	***!*			

Pass 1

Input: /σ(σσ)σ/	Parse-σ	Non-Finality	Non-Initiality	Align-σ-L
☞A: σ(σσ)(σ)	*	*		
B: (σ)(σσ)σ	*		*	*!
C: σ(σσ)σ	**!			*

Pass 2

Input: /σ(σσ)(σ)/	Parse-σ	Non-Finality	Non-Initiality	Align-σ-L
A: σ(σσ)(σ)	*!	*		
B: (σ)(σσ)(σ)		*	*	

Pass 3

The result of a four-syllable input is an output with three stresses – one on both the initial foot and the final foot, and one on the foot between them (depending on the ranking of Iamb and Trochee).

So, while the two constraints do help edge-undergeneration, they will likely also need additional constraints to restrict massive overgeneration. Substituting Parse-σ with NoParse (assign 10 violation marks for each form which consists only of unparsed syllables, assign no violation marks if the candidate does not consist only of unparsed syllables) helps to get the output patterns down from 1764 to 739 for SCA. Out of these 739 there are 289 unique stress patterns. This is on par with PCA and close to PGA. The number of patterns and unique patterns for SGA and SCA combined with Parse-σ and NoParse are given in (143).

(143)

	Non-Finality+Non-Initiality + Parse-σ	Non-Finality+Non-Initiality + NoParse
Categorical	1764 (786 unique)	739 (289 unique)
Gradient	2155 (1066 unique)	311 (167 unique)

Though NoParse has no ground in previous literature, it does a good job of restricting the amount of emerging stress patterns compared to Parse-σ, presumably because it goes unviolated after the first pass if a foot is built. As shown in chapter 3, interactions between Parse-σ and other constraints sometimes lead to unwanted stress patterns (category I), or a failure to generate stress patterns because a high-ranked Parse-σ dictates the need for binary feet to parse as many syllables as quickly as possible.

4.2.2 Consistent/persistent generation

The second cause of undergeneration for Serial OT is **consistent/persistent generation**. Serial OT has a one-track mind and will keep doing what it’s doing. Consistent/persistent generation leads to undergeneration, compared to Parallel OT, in categories B, D, E, F, G, H and I. However, not all of these undergenerations involve attested patterns. The languages that SOT cannot generate because of consistent/persistent generation are Indonesian (category D), Kashaya (category E), Estonian, Chugach Alutiiq (both category F), Cayuvava, Gilbertese and Sentani (all category H). I will discuss them below, starting with the latter three category H languages.

The failure to generate any strictly ternary-only patterns (category H) stems from the ternary ILT foot needing a binary foot as a base to build a dependent. With a constraint ranking that generates ILT feet, an eight-syllable input will result in two ILT feet being built in the first four passes (two per ILT foot). Then, for a strict ternary-only pattern, the remaining two syllables need to stay unparsed. However, since binary feet have been constructed before (in pass 1 and 3), it will happen again and indeed the results in chapter 3 show that the remaining syllables will be formed into a foot. A seven-syllable input, however, can leave a final syllable unparsed if Align-Un-L/R are ranked above Align-σ-L/R and Parse-σ, which avoids a unary foot from being built. Non-Finality can be used to prevent parsing of the final syllable, and if a unary foot is prevented from being built at the same time, then an eight-syllable input can have ((σ’σ)σ)((σ,σ)σ)σσ as output (note that this was not generated in chapter 3 at all). The entire output pattern looks as in (144).

(144)

(‘σ)σ	(‘σσ)σ	((‘σσ)σ)σ	((‘σσ)σ)σσ	((‘σσ)σ)(,σσ)σ	((‘σσ)σ)((,σσ)σ)σ	((‘σσ)σ)((,σσ)σ)σσ
Uσ	**Bσ**	**Tσ**	**Tσσ**	**TBσ**	**TTσ**	**TTσσ**

Even though the six-syllable output is not strictly ternary-only, the underlying stress pattern is: the dependent syllable does not change the stress placement. So as long as there is no difference between an unparsed syllable and a dependent syllable this can be considered a strictly ternary-only stress pattern. Note that in case of an iambic pattern, the stress placement in a two-syllable output remains the same because there is no binary foot there, only a unary – leading to inconsistent stress placement (stress is on the first syllable only in a two-syllable input, and on the second syllable for every longer input).

In category D, Serial OT cannot generate Indonesian, a **B*(T)B** pattern, and adding Non-Finality or Non-Initiality to the constraint set does not help. A seven-syllable input for Indonesian requires a (σσ)((σσ)σ)(σσ) output. As said before, Serial OT has a one-track mind and will keep doing what it started. At the first pass through EVAL, a binary foot is created at the left edge. At the second pass through EVAL the choice is between adding a binary foot and expanding the first binary foot to an ILT foot. For the Indonesian pattern building an extra binary foot is required, and at the third pass this second binary foot must be expanded to an ILT foot. The process is shown in (145).

(145)

(σσ)σσσσσ	(σσ)(σσ)σσσ	(σσ)((σσ)σ)σσ	(σσ)((σσ)σ)(σσ)

Due to consistent/persistent generation, however, this process will never take place for Serial OT. After adding a second binary foot Serial OT will keep generating binary until it is forced to stop by a lack of unparsed syllables. At that point, if one last unparsed syllable remains an ILT foot can be created as in category D in chapter 3 (or a unary foot can be built as in category C).

In categories E and F, problematic patterns for Serial OT stemming from consistent/persistent generation are **T(B*)(U)** and **T*(B)(B)**, the former is attested by Kashaya, and the latter by Estonian and Chugach Alutiiq. Both are impossible to generate with Serial OT because they start with an ILT foot and then reject building additional ILT feet, opting for binary feet instead, and even a unary foot in the case of Kashaya.

Adding constraints will not fix the undergenerations caused by consistent/persistent generation, although, as shown in (143) it may flesh out the typology. Indeed, its stubbornness it is more of a feature: Serial OT does not look at what is coming, but rather just does its job locally. In contrast, Parallel OT will fit an ILT foot at the best possible place if an ILT foot is needed (for example, in the middle of the 7-syllable form in Indonesian).

A case of where edge-undergeneration and consistent/persistent generation collide is that Serial OT cannot generate Winnebago. Winnebago is the only language that comes from a pattern (**T*(B/σ)**) that Serial OT generates 8 of, instead of the 16 Parallel OT can generate (so it is one of the 8 that Parallel OT does generate, while Serial OT does not). Ternary feet are involved, and thus Serial OT is unable to generate half the patterns that Parallel OT does because of edge-undergeneration. In Winnebago, every third syllable is stressed, and if the word ends in two unparsed syllables then the second of those is stressed too, as in (146) below.

(146)

(σ’σ)	(σ(σ’σ))	(σ(σ’σ))σ	(σ(σ’σ))(σ,σ)	(σ(σ’σ))(σ(σ,σ))	(σ(σ’σ))(σ(σ,σ))σ	(σ(σ’σ))(σ(σ,σ))(σ,σ)
B	**T**	**Tσ**	**TB**	**TT**	**TTσ**	**TTB**

While Serial OT can generate similar patterns, this one is out of reach. The four-syllable form, for instance, has the binary foot required to construct the ILT foot comprising the two middle syllables. Non-Finality can facilitate constructing the four-, six- and seven-syllable form by building ILT feet from right to left by leaving the rightmost syllable unparsed. The five- and eight-syllable forms, however, cannot be constructed. They cannot be built right-to-left because of consistent/persistent generation: the constraint ranking would lead to a binary stress system when an extra binary foot was added in the second pass through EVAL – Serial OT keeps doing the same thing. Constructing the stress pattern from left to right is not possible due to edge-undergeneration. Note that the leftmost ILT foot can be explained by Non-Initiality (creating a binary foot one syllable away from the edge, and then later making the syllable between the foot and the edge the dependant of the ILT foot), but not the second, for two reasons. First, there is no constraint which encourages an unparsed syllable between feet. Second, the combination of Align-σ-L and Align-σ-R causes any unparsed syllables caught between feet to receive twice the amount of violation marks in case of categorical constraints, as in (17). For gradient constraint, there is little to no advantage to generate a foot at the opposite edge of the initial foot either, even though the amount of violation marks does not increase. However, when one of Align-σ-L/R dominates the other, it is still advantageous to align feet so that the dominant constraint remains unviolated. If both are in the same stratum, it does not make a difference in the amount of violation marks for that stratum. However, evidenced by chapter 3, no unparsed syllables appear between feet.

(147)

σσσσσσ	Align-σ-L categorical	Align-σ-R categorical	Align-σ-L gradient	Align-σ-R gradient
(σσ)σσ(σσ)	**	**	**	**
(σσ)(σσ)σσ	**		****	
σσ(σσ)(σσ)		**		****
(σσ)σ(σσ)σ	**	*	***	*

4.2.3 Parallel OT

The two sections above show shortcomings of Serial OT since most undergeneration happens for Serial OT. However, PGA also undergenerates one language: Indonesian. The table in (84) shows the cause of this undergeneration: the distance-sensitivity generally associated with GA constraints. This sensitivity is not so severe, however, that it will undergenerate any other attested stress patterns from its typology. Also note that the Indonesian pattern is controversial (Kager 2001, Hyde 2008) and may only be the case in loan words, where the stresses may be loaned along with the words.

4.2.4 Interim summary

Undergeneration in Serial OT can be captured by two restrictive mechanisms: edge-undergeneration and consistent/persistent generation. Edge-undergeneration means that Serial OT (with the current constraint set) will always generate feet at the edges of words and previously built feet, leaving no room to expand to ILT feet towards one of the word edges. Adding Non-Finality and Non-Initiality to the constraint set helps Serial OT overcome this obstacle. However, adding two constraints to the set will inflate the number of patterns generated. Substituting Parse-σ for NoParse helps curb this inflation, and the combination of NoParse, Non-Finality, Non-Initiality and gradient constraints produces a typology with number on par with Parallel OT.

Consistent/persistent generation leads to undergeneration as well. While changes to the constraint set can aid in mitigating edge-undergeneration, consistent/persistent generation can be considered a feature of Serial OT: due to the inability to look ahead the foot generation process is highly local.

While Serial OT has some serious issues regarding undergeneration, this does not appear to be the case for PCA and PGA, at least compared to each other and to Serial OT. PGA only undergenerates the controversial Indonesian pattern.

After the discussion on undergeneration it is now time to look at overgeneration.

4.3 Overgeneration

In total, 23 of 33 BTU-patterns are unattested. The breakdown per generation method is shown in (148) below.

(148)

Framework	Number of BTU-patterns	Number of unattested BTU-patterns
PCA	22	12
PGA	21	12
SOT	11	5

Overgeneration is a more serious issue for Parallel OT than for Serial OT. However, not all overgeneration is bad – only pathological overgeneration should be considered unwanted. Martínez-Paricio & Kager (to appear) found that some of the patterns in their PCA typology constituted a case of pathological overgeneration. They describe six kinds of overgeneration that are considered pathological. Only ‘three-syllable exceptionality’ and ‘non-directionality’ are included in the PCA typology. I will argue that there is additional pathological overgeneration in the PCA typology, and show more pathological overgeneration in the other typologies as well.

Three-syllable exceptionality occurs in patterns with binary feet where a ternary ILT foot is created in the trisyllabic form, occurring in categories B and C. I will discuss the pattern in category B first: **B*(σ); T in 3σ. T**here is an unexpected ternary foot in an otherwise binary system in this stress pattern. The trisyllabic input is parsed as a ternary ILT foot. The consequences are minor: only one stress is present when one stress is expected. Furthermore, while the ternary foot can change the placement of stress, for half the patterns it does not, resulting in stress placement as in a strictly binary system. In (149), the three- and five-syllable forms for each of the 16 **B*(T); T in 3σ** patterns generated are shown. The five-syllable forms serve as illustration to show that the 16 patterns are all unique, and how stress placement occurs in that particular pattern.

(149) Three- and five-syllable forms of all 16 **B*(σ); T in 3σ** patterns.

((‘σσ)σ) / (‘σσ)(,σσ)σ	((‘σσ)σ) / (,σσ)(‘σσ)σ	((σ’σ)σ) / (σ’σ)(σ,σ)σ	((σ’σ)σ) / (σ,σ)(σ’σ)σ
(σ(‘σσ)) / (‘σσ)(,σσ)σ	(σ(‘σσ)) / (,σσ)(‘σσ)σ	(σ(σ’σ)) / (σ’σ)(σ,σ)σ	(σ(σ’σ)) / (σ,σ)(σ’σ)σ
((‘σσ)σ) / σ(‘σσ)(,σσ)	((‘σσ)σ) / σ(,σσ)(‘σσ)	((σ’σ)σ) / σ(σ’σ)(σ,σ)	((σ’σ)σ) / σ(σ,σ)(σ’σ)
(σ(‘σσ)) / σ(‘σσ)(,σσ)	(σ(‘σσ)) / σ(,σσ)(‘σσ)	(σ(σ’σ)) / σ(σ’σ)(σ,σ)	(σ(σ’σ)) / σ(σ,σ)(σ’σ)

The shaded patterns are the patterns where the stress in the three-syllable form does not fall on the syllable one would expect it to fall from observing the five-syllable form. The first column in the second row, for instance, shows stress falling on the first syllable in the five-syllable form, but on the second syllable in a three-syllable form. So, while both PGA and PCA generate this pattern, only half of the results are problematic. Note that SOT only generates eight patterns in this system – only the ones in the non-shaded cells in (149).

Three-syllable exceptionality also occurs in **B*(U); T in 3σ** in category C. In this case, the trisyllabic form has only primary stress where an additional secondary stress would be expected. Martínez-Paricio & Kager (to appear) argue that the lack of attestation can be a result of perceptual factors. The difference between stressed and unstressed is difficult to perceive on word-final syllables due to final lengthening, which is more prominent on shorter words (Hayes 1995, Lunden 2014). A learner (or linguist) may not perceive the stress on the final syllable and interpret the word as having primary stress only. As a result, a pattern that should be **B*(U)** can be interpreted as being **B*(U); T in 3σ**.

Even if the metrical structure is inconsistent between forms of length *n* and *n+i* and the pattern could thus be seen as pathologic, the reason can be explained by faulty perception and learnability.

Non-directionality concerns an inconsistent directionality of iterative footing. Non-directional forms can be found in categories D, E, F, G and I. In (150) an example of non-directionality is given: **(B)T*(U)**.

(150)

(B)T*(U)
(σσ)
(σσσ)
(σσσ)(σ)
(σσ)(σσσ)
(σσσ)(σσσ)
(σσσ)(σσσ)(σ)
(σσ)(σσσ)(σσσ)

The binary feet are anchored to the left edge of the word, while unary feet are anchored to the right: there is an inconsistent directionality. Elenbaas & Kager (1999) judge any kind of directional pattern to be pathological, but this may be a premature conclusion. Firstly, learnability can come into play: a pattern could still be a possible language, but harder to learn, for instance due to variability in the position of primary stress in words of different lengths (Staubs 2014). Reduced learnability may lead to reduced representation. Primary stress placement is generally independent of other stress placement (Van der Hulst 1996), meaning that it can usually be expressed as stress on a certain syllable: initial, final, pre-final etc. instead of the rightmost or leftmost placed stress. Goedemans (2010) shows that parsing direction is closely related to primary stress placement, shown in Staubs (2014) and replicated here in (151).

(151)

	Left-to-right	Right-to-left
Left primary	63	12
Right primary	27	57

$\chi2 = 38.1, p < 0.05$

Since the constraints in this thesis are completely symmetrical (all constraints have a left and right variant), this learnability component is not built into the constraints and so the typology cannot reflect this component. A second learnability component is that many of the stress patterns only differ minutely from others – it could be the case, for instance, that only the eight-syllable forms are different in patterns X and Y. If this is the case it may be possible that no eight-syllable word ever comes along to differentiate between pattern X and Y. Table (152) below shows the amount of unique patterns - including foot structure - per method per maximum amount of syllables. The number of unique patterns – stress only, without taking foot structure into account – is shown in parentheses.

(152) The number of unique patterns with different syllable caps

	5 syllables	6 syllables	7 syllables	8 syllables
PCA	140 (112)	252 (208)	300 (264)	316 (288)
PGA	156 (118)	284 (236)	300 (264)	300 (264)
SOT	67 (47)	83 (47)	83 (55)	83 (63)

The table in (152) shows that languages that have words (or word stems) of maximally five syllables have noticeably fewer stress patterns than languages that allow longer words. It is likely that a typology that goes to nine syllables will see an increase in total stress patterns. Stanton (2014) shows that English words rarely have more than five vowels, corresponding roughly to five syllables. Other languages may have up to twelve syllables. The median learner of a language is exposed to words with up to six vowels, where the six-vowel words make up 1% of the total input. Child-directed speech, assuming children are the primary learners, may be even simpler. So, stress patterns for which only the forms with eight and seven syllables are different from other stress patterns may be harder to learn. With this in mind, scrapping the seven- and eight-syllable outputs from the stress patterns, many unattested patterns disappear. Only four different patterns remain in the problematic category I; only one pattern remains in category F (which is attested) and two unattested patterns disappear from category E. Other unattested patterns remain.

Category I is not discussed in Martínez-Paricio & Kager (to appear), though it does have some unexpected stress placement. Additionally, category I has some unexpected stress placement. Martínez-Paricio & Kager (to appear) do not judge this category to be pathological. In these patterns, at most two feet are created before parsing stops. An eight-syllable input surfaces as (σσσ)(σσ)σσσ or (σσ)(σσσ)σσσ (only the former for Serial OT). The constraints used in their (and this) paper allow stress to fall on either the leftmost or the rightmost foot, regardless of how many unparsed syllables separate the foot from an edge. This means that for eight syllable inputs the primary stress can fall on the fourth or fifth syllable – outside the stress window of Kager (2012). This is the case for category H (strictly ternary) as well for forms of 3*n+2* syllables, with *n* being bigger than 1. A solution might be to devise a constraint that limits primary stress to a foot aligned with the word edge – the primary stress window is three syllables from each edge. This might create a problem if Non-Finality and Non-Initiality are also in the constraint set, however.

Category I and H are dissimilar for forms above eight syllables, since a strictly ternary pattern will create an extra ILT foot whenever three syllables are available, whereas category I patterns generated by PCA will only ever generate two feet. Rhythmic patterns like the ones in category I are unattested and in my eyes qualify as a pathological overgeneration. In the six pathological overgeneration categories presented by Martínez-Paricio & Kager (to appear), no category is dedicated to patterns where parsing stops after *n* syllables.

Another problematic category is **B*(σ); Tσσ in 5σ**, where in forms of five syllables only a ternary ILT foot is created. A primary and a secondary stress are expected, but only a primary stress is formed. Forms of four and six syllables do have the expected amount of stresses (two and three respectively), so this pattern can be qualified as pathological. It is only generated with PGA.

All things considered, PGA and PCA are not much different. Neither framework undergenerates, save for the exclusion of Indonesian by PGA, and neither overgenerate excessively. It appears that PCA is perhaps more robust that PGA in the sense that a Parse-σ can be added to the constraint set without resulting in pathological overgeneration, which is the case for PGA, where PGAP produces extra patterns in category I.

Chapter 5. Conclusions

The research topic of this thesis was to find out in what way different versions of Optimality Theory, specifically the Serial and Parallel versions and the gradient and categorical constraints, and any interactions between them, shape the factorial typology. To do so, I started with the typology that was presented in Martínez-Paricio & Kager (to appear) and generated additional typologies to contrast and compare.

In chapter 2, I have shown some of the literature showing positives and negatives of both Serial OT and Parallel OT. I have done the same for gradient constraints and categorical constraints. In chapter 3, I have showed the typologies resulting from all different permutations of Parallel and Serial, and gradient and categorical. I have divided the results into rhythmic categories and shown why some methods do or do not generate certain patterns. By doing so, I have shown that the issues reported in the literature such as the midpoint pathology, odd-parity input-problem and the drifting foot problem do not occur with the constraint set proposed by Martínez-Paricio & Kager (to appear). They have already remarked upon this for their PCA typology, and I have shown that the constraints are robust enough to also be able to be used with gradient constraints and Serial OT, even though Serial OT undergenerates.

In chapter 4, I have discussed the typologies presented in chapter 3, and shown where there is undergeneration and overgeneration. Serial OT suffers from two kinds of undergeneration. Edge-undergeneration occurs because binary feet are always built aligned with the word edge or a foot edge, leaving only room to expand to ILT feet in one direction. This is the reason Serial OT with the constraints as used in this thesis does not generate languages with stress on the third syllable, such as Macedonian. A possible fix is introducing Non-Finality and Non-Initiality as additional constraints. Using NoParse instead of Parse-σ can hem in the additional stress patterns that emerge with Parse-σ, for both gradient and categorical constraints.

I have also shown that there is little pathological overgeneration. I have suggested a constraint that limits primary stress to a word edge-aligned foot to limit primary stress from occurring on fourth and fifth syllables, thus reigning in some additional overgeneration.

From the four typologies presented, both PGA and PCA show a limited amount of overgeneration, with few truly pathological patterns. Additionally, the only undergeneration that appears is Indonesian in PGA, which is a controversial pattern. PCA has shown to be more robust, since adding Parse-σ to constraint set does not result in pathological overgeneration. Serial OT with the current constraint set has too much undergeneration. A topic of further research can be to find in what way additional constraints can stop Serial OT from undergenerating while not creating an unreasonably large typology.

This thesis can serve as a jumping-off point for further research. For instance, a similar account can be made for quantity-sensitive stress to further advance the discussion between the two OT methods and the two types of constraints. Obviously, additional constraint will be necessary to deal with the difference between light and heavy syllables. Another continuation of the topic presented in this thesis could be to explore the typologies created by substituting Parse-σ for NoParse and adding Non-Finality and Non-Initiality to the constraint set for Serial OT (and possible Parallel OT), and possibly the constraint to limit primary stress to word edge-aligned feet (this may be problematic with Non-Finality and Non-Initiality). A further topic for analysis could be to try to introduce the learnability component described in Staubs (2014) into the constraint set somehow.

Bibliography

Bane, M., & Riggle, J. (2009). The typological consequences of weighted constraints. In *Proceedings from the Annual Meeting of the Chicago Linguistic Society* (Vol. 45, No. 1, pp. 13-27). Chicago Linguistic Society.

Bıró, T. (2003). Quadratic alignment constraints and finite state Optimality Theory. In *Proceedings of the Workshop on Finite-State Methods in Natural Language Processing (FSMNLP), 10th Conference of the European Chapter of the ACL, Budapest* (pp. 119-126).

Blevins, J., & Harrison, S. P. (1999). Trimoraic feet in Gilbertese. *Oceanic Linguistics*, 203-230.

Boersma, P., & Hayes, B. (2001). Empirical tests of the gradual learning algorithm. *Linguistic inquiry*, *32*(1), 45-86.

Buckley, E. (1997). Optimal iambs in Kashaya. *Rivista di linguistica*, *9*, 9-52.

Buckley, E. (2009). Locality in metrical typology. *Phonology*, *26*(03), 389-435.

Buckley, E. (2014, March). Kashaya Extrametricality and Formal Symmetry. In*Proceedings of the Annual Meetings on Phonology* (Vol. 1, No. 1).

Chomsky, N. (1965). *Aspects of the theory of syntax*. Cambridge, Mass.: MIT Press.

Cohn, A. C. (1989). Stress in Indonesian and bracketing paradoxes. *Natural language & linguistic theory*, *7*(2), 167-216.

Cowan, H. K. J. (2013). *Grammar of the Sentani language* (Vol. 47). Springer.

Das, S. (2001). *Some aspects of the phonology of Tripura Bangla and Tripura Bangla English* (Doctoral dissertation).

Dresher, B. E., & Lahiri, A. (1991). The Germanic foot: metrical coherence in Old English. *Linguistic inquiry*, 251-286.

Eisner, J. (1997). What constraints should OT allow? Paper presented at *71st Annual Meeting of the Linguistic Society of America*, Chicago. Available as ROA-204 from the Rutgers Optimality Archive.

Eisner, J. (1997). FootForm decomposed: Using primitive constraints in OT. *Proceedings of SCIL VIII*. Cambridge, MA. 115-143.

Eisner, J. (2000). Directional constraint evaluation in Optimality Theory. In*Proceedings of the 18th conference on Computational linguistics-Volume 1* (pp. 257-263). Association for Computational Linguistics.

Elenbaas, N., & Kager, R. (1999). Ternary rhythm and the lapse constraint.*Phonology*, *16*(03), 273-329.

Ellison, T. M. (1994). Phonological derivation in optimality theory. In*Proceedings of the 15th conference on Computational linguistics-Volume 2* (pp. 1007-1013). Association for Computational Linguistics.

Furby, C. E. (1974). Garawa Phonology. *Papers in Australian Linguistics* 7. 1-11

Goedemans, R. (2010). A typology of stress patterns. Van der Hulst, Harry, Rob Goedemans & Ellen van Zanten (eds.), *A survey of word accentual systems in the language of the world*, Mouton de Gruyter, Berlin, 647–666.

Gonzalez, C. (2005). Phonologically-conditioned allomorphy in Panoan: towards an analysis. In Heinz, J., Martin, A., and Pertsova, K. (eds*.) UCLA Working Papers in Linguistics 11: Papers in Phonology* 6. 39-56.

Halle, M. (1990). Respecting metrical structure. *Natural Language & Linguistic Theory*, *8*(2), 149-176.

Halle, M. & Vergnaud, J. (1987). An essay on stress. Cambridge, MA: MIT Press.

Hawkins, W. N. (1950). Patterns of vowel loss in Macushi (Carib). *International Journal of American Linguistics*, *16*(2), 87-90.

Hayes, B. (1995). *Metrical stress theory: Principles and case studies*. University of Chicago Press.

Hayes, Bruce, Tesar, B & Zuraw, K (2003). OTSoft 2.3.3. Software package, http://www.linguistics.ucla.edu/people/hayes/otsoft

Hercus, L. (1969). *the languages of Victoria: A Late Survey (Parts I and II).* Canberra: Australian Institute of Aboriginal Studies.

Hercus, L. (1986). *Victorian Languages: a late survey.* Canberra: Australian National University.

Hewitt, Mark S. (1992). Vertical maximization and metrical theory. PhD dissertation, Brandeis University, Waltham, MA.

Hualde, J. I. (1998). A gap filled: postpostinitial accent in Azkoitia Basque.*Linguistics*, *36*(1), 99-118.

Hyde, B. (2007). Non-finality and weight-sensitivity. *Phonology*, *24*(02), 287-334.

Hyde, B. (2008). Alignment continued: distance-sensitivity, order-sensitivity, and the Midpoint Pathology. *MS., Washington University*.

Hyde, B. (2012a). Alignment constraints. *Natural Language & Linguistic Theory*,*30*(3), 789-836.

Hyde, B. (2012b). The odd-parity input problem in metrical stress theory.*Phonology*, *29*(03), 383-431.

Itô, J., & Mester, A. (1992). Weak layering and word binarity. *Ms., University of California, Santa Cruz*.

Kager, R. (1995). Ternary rhythm in alignment theory.Ms, University of Utrecht. Available as ROA-35 from the Rutgers Optimality Archive.

Kager, R. (1996). On affix allomorphy and syllable counting. Ursula Kleinhenz (ed.), *Interfaces in phonology*. Berlin: Akademie Verlag. 155-171.

Kager, R. (1999). *Optimality Theory*. Cambridge: Cambridge University Press.

Kager, R. (2001). Rhythmic directionality by positional licensing. Fifth HIL Phonology Conference (HILP 5). University of Potsdam.

Kager, R. (2012). Stress in windows: Language typology and factorial typology.*Lingua*, *122*(13), 1454-1493.

Kager, R. & Martínez-Paricio, V. (2013). Ternary stress in Parallel and Serial OT: consequences for representations. Poster presented at Phonology 2013, University of Massachusetts.

Key, H. (1961). Phonotactics of Cayuvava. *International journal of American linguistics*, 143-150.

Krauss, M. (1985). *Yupik Eskimo Prosodic Systems: Descriptive and Comparative Studies. Alaska Native Language Center Research Papers No. 7*. Alaska Native Language Center, University of Alaska.

Leer, J. (1985a). Prosody in Alutiiq. in Krauss (1985). 77-134.

Leer, J. (1985b). Evolution of prosody in the Yupik languages. In Krauss (1985). 135-158.

Leer, J. (1985c). Toward a metrical interpretation of Yupik prosody. In Krauss (1985). 159-173.

Levin, J. (1985). Evidence for ternary feet and implications for a metrical theory of stress rules. Ms, University of Texas at Austin

Lunden, S.L.A. (2014). Motivating final stress lapse. Presentation at Conference on Stress and Accent, University of Leinden, August 16, 2014.

Martínez-Paricio, V. (2013). An exploration of minimal and maximal metrical feet. PhD dissertation, University of Tromsø.

Martínez-Paricio, V. & Kager, R. (to appear). Non-intervention constraints and the binary-to-ternary rhythm continuum.

McCarthy, J. J. (1982). Prosodic structure and expletive infixation. *Language*, 574-590.

McCarthy, J. J. (2000). Harmonic Serialism and parallelism. *Proceedings of the North east Linguistics Society*. Eds. Masako Hirotani, Andries Coetzee, Nancy Hall and Ji-yung Kim. Amherst, MA: GLSA. 501-524

McCarthy, J. J. (2003). OT constraints are categorical. *Phonology*, *20*(01), 75-138.

McCarthy, J. J. (2006). Restraint of analysis. In Eric Bakovic, Junko Ito and John J. McCarthy (eds.) *Wondering at the natural fecundity of things: essays in honor of Alan Prince*. Santa Cruz: Linguistics Research Center. 213-239. Also in Blaho et al. (eds.). 203-231.

McCarthy, J. J. (2007). *Hidden Generalizations: Phonological Opacity in Optimality Theory*. London: Equinox.

McCarthy, J. J. (2008). The serial interaction of stress and syncope. *Natural Language & Linguistic Theory*, *26*(3), 499-546.

McCarthy, J. J. (2010). An introduction to harmonic serialism. *Language and Linguistics Compass*, *4*(10), 1001-1018.

McCarthy, J.J. & Prince, A. (1986). *Prosodic Morphology*. Ms. University of Massachusetts, Amherst & Brandeis University.

McCarthy, J. J. & Prince, A. (1993). Generalized Alignment. In Geert Booij & Jaap van Marle (eds.) Yearbook of morphology 1993. Dordrecht: Kluwer. 79-153.

Miner, K (1979). Dorsey's law in Winnebago-Chiwere and Winnebago accent. *International Journal of American Linguistics* 45. 25-33.

Mürk, H. W. (1991). The structure and development of Estonian morphology. Doctoral dissertation, Indiana University.

Prince, A. S. (1980). A metrical theory for Estonian quantity. *Linguistic Inquiry*, 511-562.

Prince, A. S. (1983). Relating to the grid. *Linguistic inquiry*, 19-100.

Prince, A.S. & Smolensky, P. (1993). Optimality Theory: constraint interaction in generative grammar. Ms, Rutgers University & University of Colorado, Boulder. Available as ROA-537 from the Rutgers Optimality Archive.

Pruitt, K. (2010). Serialism and locality in constraint-based metrical parsing.*Phonology*, *27*(03), 481-526.

Pruitt, K. (2015). HS in OT-Help 2. https://sites.google.com/site/katpru/research/hs-in-ot-help-2

Rice, C. (1988). Stress assignment in the Chugach dialect of Alutiiq. *CLS* 24. 304-315

Rice, C. (1990). Pacific Yupik: inplications for metrical theory. *Coyote Papers*. Tuscon, Arizona: Department of Linguistics, University of Arizona.

Rice, C. (1992). *Binarity and ternarity in metrical theory: parametric extensions* (Doctoral dissertation, University of Texas at Austin).

Rice, C. (2007). The roles of GEN and CON in modeling ternary rhythm.*Freedom of analysis*, 233-255.

Selkirk, E. O. (1980). The role of prosodic categories in English word stress.*Linguistic inquiry*, 563-605.

Stanton, Juliet (2014). Learnability shapes typology: the case of the midpoint pathology. Paper presented at the Annual Meeting on Phonology, MIT.

Staubs, R. (2014). Learning and the position of primary stress. In *Proceedings of the 31st West Coast Conference on Formal Linguistics* (pp. 428-437).

Staubs, R., Becker, M., Potts, C., Pratt, P., McCarthy J.J. & Pater, J. (2009). OT-Help 2.0. Software package, http://web.linguist.umass.edu/~OTHelp/

Tesar, B. (1996, June). Computing optimal descriptions for Optimality Theory grammars with context-free position structures. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics* (pp. 101-107). Association for Computational Linguistics.

Van der Hulst, H. (1996). Separating primary and secondary accent. Goedemans, Rob, Harry van der Hulst & Ellis Visch (eds.), *Stress patterns of the world*, Holland Academic Graphics, The Hague.

Wolf, M. (2008). Optimal interleaving: serial phonology-morphology interaction in a constraint-based model. PhD thesis, UMass, Amherst.

Wolf, M. (2013). Candidate chains, unfaithful spell-out, and outwards-looking phonologically-conditioned allomorphy. *Morphology*, *23*(2), 145-178.