Representing and learning stress in a MaxEnt framework

Seung Suk Lee, Joe Pater, & Brandon Prickett
University of Massachusetts Amherst

1 Introduction

This paper summarizes some of the ideas and results of Lee et al. (2024), and adds an in-depth presentation of the software package 'SoftStress' (Lee & Pater, 2024) that we have developed to learn and solve weighted constraint grammars with hidden structure.

We adopt Maximum Entropy Grammar (Goldwater & Johnson, 2003) as our general framework for learning, which is a weighted constraint grammar that defines a probability distribution over a set of candidates in a tableau. Learning consists of a gradual update of the constraint ranks/updates, based on the difference between the learner's observed probability distribution in the learning data and the learner's estimated probability distribution given the current weights.

Using this learning theory, we compare theories of grammar that differ both in the assumptions of how a given linguistic form is represented (i.e., candidates in the tableaux) and in the set of constraints used to evaluate these representations. In particular, we compare two approaches to the representation of metrical structure: feet vs. grids.

These two representations fundamentally differ in the sense that feet introduce the so-called 'problem of hidden structure in learning' (Tesar & Smolensky, 2000) while grids do not, which has been one of the central motivations for assuming grid-based representation for metrical structure (Gordon, 2011:163). In a foot-based theory of stress, an overt form that the learner is presented with may be compatible with multiple prosodifications. For instance, a sequence of three light syllables where the middle syllable bears the primary stress, represented with the notation [L L1 L] following Tesar & Smolensky (2000), is compatible with two hidden representations (HR): a left-aligned iamb [(L L1) L] and a right-aligned trochee [L (L1 L)]. We use the term 'SR' to refer to the overt form (e.g., [L L1 L]) that the learner is presented with, and the term 'HR' to refer to the hidden form that the learner needs to infer (e.g., [(L L1) L] or [L (L1 L)]). It has been argued that the presence of hidden representation potentially poses a learning challenge since learners need to infer the correct prosodifications based on the probability distribution over the SRs in order to learn the pattern (Gordon, 2011). More recently, Pater & Prickett (2022) and Lee et al. (2023) have shown that stress systems with hidden structure can be successfully learned using a MaxEnt learning approach that computes the probability of the overt form by summing all possible hidden representations that are consistent with the overt form, which is first proposed in Pater et al. (2012).

To continue this line of research, we have developed a new software package named 'SoftStress' (Lee & Pater, 2024), which we present in §2. The software package is equipped with a MaxEnt hidden structure learner and a Linear Programming solver. The MaxEnt hidden structure learner is the one used in Pater & Prickett (2022) and Lee et al. (2023) and is also available as a stand-alone software pacakge in Prickett & Pater (2024). We demonstrate how the learner works with a concrete example in §2.1. The Linear Programming solver is a python implemented version of OT-Help (Staubs et al., 2010) based on the proposals made in Potts et al. (2010). It finds the set of constraint weights that make the observed overt forms optimal across multiple

* We thank all the people at the UMass Sound Workshop, especially Gaja Jarosz, anonymous AMP reviewers, and participants in AMP 2024. This research was supported by NSF grant BCS-2140826.

Proceedings of *AMP 2023/2024*

tableaux given the input pattern. In order to find if a solution exists for a given pattern in the presence of hidden structure, its predecessor, OT-Help (Potts et al., 2010) needs to first generate the factorial typology, and then check if the pattern in question exists in the typology. The current implementation can iterate over all possible combinations of hidden representations that are compatible with the observed SR, such that all possible sets of analyses may be found for the input pattern, without having to generate the factorial typology. We introduce the solver in §2.2.

In the rest of the paper, we show how these tools may be used to compare theories of grammar, i.e. foot-based vs. grid-based representations of stress, in terms of learnability of a typology of attested stress systems, which we export from StressTyp2 (Goedemans et al., 2015). We adopt the foot-based constraints from Tesar & Smolensky (2000), which are based on proposals from Prince & Smolensky (1993/2004) and McCarthy & Prince (1993). The grid-based constraints are the ones proposed in Gordon (2002) to represent the quantity-insensitive patterns, with the addition of WEIGHT-TO-STRESS to represent the quantity-sensitive patterns. We also test on slightly different versions of some of the constraints in these constraint sets. While the current paper only provides descriptive results, we refer more interested readers to Lee et al. (2024) where we provide a more thorough description of our results of investigating the constraint sets and some of the individual patterns. We conclude the paper with dicussions and future directions.

2 SoftStress: a new python program to learn/solve weighted constraint grammars with hidden structure

This section presents a new python program developed to learn and solve weighted constraint grammars with hidden structure, which we call 'SoftStress' (Lee & Pater, 2024). It is equipped with a MaxEnt hidden structure learner (Prickett & Pater, 2024) and an LP solver based on the proposal in Potts et al. (2010).

2.1 *Hidden structure learning with MaxEnt* In this section, we show with a simplified example how our MaxEnt hidden structure learner works. The learner is also available as a stand-alone program at `https://github.com/blprickett/Hidden-Structure-MaxEnt` (Prickett & Pater, 2024).

In the MaxEnt learning we adopt, the constraint weights are updated based on the violation vector obtained using the probabilities assigned to all of the consistent candidates for the winning surface representations (see Pater et al., 2012). We demonstrate how this is done with a concrete example in Table 1. In this example, the learner is presented with two learning data: [L L1] and [L L1 L]. To make the example simpler, we limit our candidates to be ones with strictly binary feet. Table 1 consists of two tableaux. The top tableau shows possible surface representations (SRs) for the Underlying representation (UR) [L L] in the first column (1a and 1b), and their prosodified representations (i.e., the hidden rerpesentations or HRs) in the second column. The bottom tableau shows all three possible SRs for [L L L] in the first column (2a-c). Among these SRs, the candidate in 2a is compatible with two HRs in (2a-i) and (2a-ii). The right most column with the heading 'observed probability (p_{obs})' indicates that the observed SRs that we want the learner to learn are [L L1] and [L L1 L].

	SR	Weight HR	1 IAMB	1 NONFIN	1 FNF	3 AFR	$\mathcal{H}$	p_{exp}	p_{obs}
1a.	[L L1]	[(L L1)]	0	-1	-1	0	-2	0.5	1
1b.	[L1 L]	[(L1 L)]	-1	-1	0	0	-2	0.5	0
2a-i.	[L L1 L]	[(L L1) L]	0	0	-1	-1	-4	0.06	1
ii.		[L (L1 L)]	-1	-1	0	0	-2	0.44	
2b.	[L1 L L]	[(L1 L) L]	-1	0	0	-1	-4	0.06	0
2c.	[L L L1]	[L (L L1)]	0	-1	-1	0	-2	0.44	0

Table 1: Two tableaux for [L L] and [L L L]. The observed SRs for this pattern are [L L1] and [L L1 L].

The HRs are evaluated by the four weighted constraints: ALLFEET-R (AFR), NONFIN, IAMB, and FOOTNONFIN (FNF). These constraints are from the constraint set proposed in Prince & Smolensky (1993/2004) and McCarthy & Prince (1993); definitions appear in §3.2. To put in simple words, ALLFEET-R requires the final syllable to be footed, while NONFIN requires the final syllable to be unparsed. IAMB and FOOTNONFIN conflict on whether the foot is right-headed or left-headed. In order to get both [L L1]

and [L L1 L] to win in their respective tableau, the constraints IAMB and NONFIN need higher weights compared to the other two, thus requiring the last syllable to be unfooted and each foot to be right-headed. Constraint violations are marked with negative integers and the weight of each constraint is in the first row. The weights are chosen by hand for this particular example. The Harmony ($\mathcal{H}$) and the MaxEnt probability (or the expected probability given the current constraint weights, i.e., p_{exp}) for each HR are shown in their respective columns. The Harmony is the weighted sum of the constraint violations and the MaxEnt probability of a candidate is proportional to the exponent of its harmony (Smolensky & Legendre, 2006; Goldwater & Johnson, 2003).

$$Exp(c_i) = \sum_{hr \in t} c_i(hr) p_{exp}(hr) \tag{1}$$

$$Obs(c_i) = \sum_{hr \in t} c_i(hr) \widehat{p_{obs}(hr)} \tag{2}$$

$$\widehat{p_{obs}(hr_i)} = \frac{p_{exp}(hr_i)}{\sum_{hr \in sr} p_{exp}(hr)} * p_{obs}(sr) \tag{3}$$

To compute the weight updates in gradient descent learning algorithm, we find the dot product of the observed and the expected probability vectors over the candidates with the violation vectors in Table 1 for each constraint. These steps are described in Equations (1-2), where $Obs(c_i)$ and $Exp(c_i)$ indicate the observed and expected violation of learner for the constraint c_i, and $c_i(hr)$ indicates the number of violations that a given HR in a tableau (t) incurs for that constraint.

While the learner estimates the expected probability for each of the hidden representations, the learner only gets the observed probability distribution over the surface representation in the training data. To get the probability of the observed form, we make use of the learner's current estimates for the hidden representation to allocate the probability of a surface representation. This is described in Equation (3), where $p_{obs}(sr)$ indicates the observed probability of the SR that are compatible with the set of HRs (hr) in the denominator. For instance, in Table 1, the surface representation [L L1 L] in (2a) has the probability of 1, which needs to be allocated to the two compatible hidden representations in (i) and (ii). Since they have the estimated probabilities of 0.06 and 0.44 respectively, we normalize these probabilities using the total probability of the compatible hidden representations (i.e., 0.06+0.44). As a result, the forms in (i) and (ii) get the normalized observed probabilities of 0.12 and 0.88. To calculate the estimated observed probability of the HR forms, we multiply these normalized probabilities with the observed probability of this SR (i.e. 1). As a result, the estimated observed probabilities of the forms in (2a-i) and (2a-ii) are 0.12 and 0.88 respectively.

		IAMB	NONFIN	FNF	AFR
1. [L L]	Observed(W)	0	-1	-1	0
	Expected(L)	-0.5	-1	-0.5	0
2. [L L L]	Obs.	-0.88	-0.88	-0.12	-0.12
	Exp.	-0.5	-0.88	-0.5	-0.12
Avg.	Obs.	-0.44	-0.94	-0.56	-0.06
	Exp.	-0.5	-0.94	-0.5	-0.06
	Current weights	1	1	1	3
Update	Obs - Exp.	+0.06	0	-0.06	0
New weights (learning rate = 100)		7	1	0	3

Table 2: Probability weighted observed and expected constraint violations and the updated weights

The results of computing Equations (1-3) are presented in Table 2. These observed and expected probability weighted violation vectors represent learner's expected and observed constraint violations given

the current weights. Once we compute the observed and expected violation vectors for each tableau, we compute the average observed and expected vectors.[1]

Finally, the weight updates for the constraints is computed by subtracting the average expected violation vector from the average observed violation vector. We assume that the weights are minimally 0 as negative weights make constraints have the opposite of their intended effect.

The updates are scaled by a learning rate. Here, we use an unusually large learning rate of 100 in this example, to demonstrate the final outcome in just a few steps. If we repeat the steps explained thus far with these new constraint weights in Table 2 scaled by the learning rate of 100, it takes just one more weight update to achieve the learning success. We refer to the number of learning updates as 'learning epochs' and we define the learning success as the summed probability of the consistent HR candidates to reach 0.9 threshold (e.g., if the total probabilities assigned to [(L L1) L] and [L (L1 L)] go over 0.9 in Table 4). We use this definition of learning success throughout this paper.

Learning Epoch	IAMB	NONFIN	FNF	AFR
0 (Initial weights)	1	1	1	3
1 (New weights in Table 2)	7	1	0	3
2 (Learning succeeds)	7	45	0	0

Table 3: Weight updates by learning epoch, using the learning rate of 100

The weight updates are presented in Table 3 and the final outcome is presented in Table 4. Table 3 shows that the learner first learns that the feet should be right-headed from the initial set of weights and updates the weights accordingly (weights at Epoch 1). Once the learner assigns a high weight on IAMB, the only two viable HR candidates are [(L L1) L] and [L (L L1)] because others violate IAMB. With these new set of weights (weights found at Epoch 2), the learner then learns that the only way to make both [L L1] and [L L1 L] optimal in their respective tableau is by assigning a high weight on NONFIN and thus penalize [L (L L1)]. This leads to the final grammar in Table 4 which assigns high weights on IAMB and NONFIN while it assigns 0 weight on the other two constraints. After this update, the learner's estimated probabilities for the SRs exceed the 0.9 threshold, which we define as the learning success, and the learning process stops.

		Weight	7	45	0	0			
	SR	HR	IAMB	NONFIN	FNF	AFR	$\mathcal{H}$	p_{exp}	p_{obs}
1a.	[L L1]	[(L L1)]	0	-1	-1	0	-45	1	1
1b.	[L1 L]	[(L1 L)]	-1	-1	0	0	-52	0	0
2a-i.	[L L1 L]	[(L L1) L]	0	0	-1	-1	0	1	1
ii.		[L (L1 L)]	-1	-1	0	0	-52	0	
2b.	[L1 L L]	[(L1 L) L]	-1	0	0	-1	-7	0	0
2c.	[L L L1]	[L (L L1)]	0	-1	-1	0	-45	0	0

Table 4: The learned grammar after two learning updates

Pater & Prickett (2022) has shown that this MaxEnt hidden structure learner achieves results comparable to the state-of-the-art results by Jarosz (2013) and Jarosz (2015) on a test set of stress systems proposed by Tesar & Smolensky (2000) (see Jarosz (2019) for a review on research on this test set). Note that Pater & Prickett (2022) uses a slightly different optimization algorithm ('L-BFGS-B') to find the optimal weights, while this section describes a simpler gradient descent algorithm. The stand-alone learner 'Hidden-Structure-MaxEnt' (Prickett & Pater, 2024) is equipped with both optimization algorithms: 'L-BFGS-B' and the gradient descent.

Subsequently, Lee et al. (2023) have also shown that this learner equipped with the foot-based candidates and constraints learn the attested typology of quantity insensitive stress systems (Gordon, 2002) similarly well compared to the same learner equipped with the grid-based candidates and constraints.

[1] Note that this implies that we assume the learner observes the SR [L L1] and the SR [L L1 L] with equal probability in the learning data.

	SR	Weight HR	1 IAMB	1 NONFIN	1 FNF	10 AFR	$\mathcal{H}$	p_{exp}	p_{obs}
1a.	[L L1]	[(L L1)]	0	-1	-1	0	-2	0.5	1
1b.	[L1 L]	[(L1 L)]	-1	-1	0	0	-2	0.5	0
2a-i.	[L L1 L]	[(L L1) L]	0	0	-1	-1	-11	0	1
ii.		[L (L1 L)]	-1	-1	0	0	-2	0.5	
2b.	[L1 L L]	[(L1 L) L]	-1	0	0	-1	-11	0	0
2c.	[L L L1]	[L (L L1)]	0	-1	-1	0	-2	0.5	0

Table 5: The same learning problem as Table 1 but with a much higher initial weight on ALLFEET-R. This weight initialization results in a learning failure.

2.1.1 *Failure of hidden structure learning due to a local optimum* The steps explained in the previous section are not guaranteed to result in a learning success due to the presence of local optima, which is a known problem for learning with hidden structure. This section presents an example adapted from Boersma & Pater (2016) where learning of the stress system using the set of constraints in Table 1 fails for a particular weight initialization.

Table 5 has the same candidates and the constraints as Table 1 and has the same set of weights for the constraints, except for the weight of ALLFEET-R, which is set to be arbitrarily high (= 10). With this set of weights, the weight updates are computed, as shown in Table 6. In the last row of Table 6, it shows that the learner ends up not updating the weights given this initial set of weights. In other words, the learner is trapped in a so-called 'local optimum' in that the learning algorithm which relies on the gradient cannot access the global optimum, which leads to the learning success. Given the learning set up presented here, when there is a high initial weight of ALLFEET-R, which represents the learner's strong initial belief that the foot lies at the right edge (a high weight on ALLFEET-R), the learner cannot consider the actual solution required to learn the observed pattern.

		IAMB	NONFIN	FNF	AFR
1	Observed(W)	0	-1	-1	0
	Expected(L)	-0.5	-1	-0.5	0
2	Obs.	-1	-1	0	0
	Exp.	-0.5	-1	-0.5	0
Avg.	Obs.	-0.5	-1	-0.5	0
	Exp.	-0.5	-1	-0.5	0
Update	Obs - Exp.	0	0	0	0

Table 6: Probability weighted observed and expected constraint violations and the updated weights. The weight update process leads to no change in constraint weights.

The example shown in this section illustrates an important point to consider when we investigate the learnability of a pattern using this learner. When a pattern is learned by the learner, it means that the pattern is representable with the set of constraints used for the learning, meaning there exists a set of weights that make the observed SRs optimal. However, when it is not learned by the learner with some weight initialization, it does not always imply that the grammar is incapable of representing the pattern. Therefore, when a pattern is not learned by the learner, it still needs to be checked whether this is because the constraint set is incapable of representing the pattern or whether the learning fails due to the problem of local optima. Next, we turn to a computational method for checking the representability of a constraint set for a pattern.

2.2 *Solving an HG grammar using Linear Programming (LP)* In this paper, we call a set of weights that make the observed SRs optimal across all tableaux as a 'solution'. In §2.1, we have described how a single solution for a given pattern can be 'learned', using a specific learning algorithm that updates the weights gradually as the learner estimates the probabilities for SRs given the current set of weights.

However, this is not the only way to figure out a solution for a weighted constraint grammar. Potts et al. (2010) develop an automated method to find a solution in a deterministic Harmonic Grammar (HG) using Linear Programming (LP). While we refer to the weights found by the MaxEnt hidden structure learner as being 'learned', we refer to the weights found by the LP method as being 'solved'.

A solution solved using LP gives a correct candidate in each tableau the highest Harmony. Such an HG solution that makes the observed candidates optimal can also make the probabilities of those candidates be arbitrarily close to 1, if MaxEnt probabilities were to be computed, by scaling the constraint weights by a constant of sufficient magnitude. See Pater (2016:§5) and Anttila & Magri (2018) for relevant discussions.

Potts et al. (2010) show that HG tableaux can be mathematically formulated in terms of a series of inequalities, which is a Linear Programming (LP) problem, each of which indicates the harmony relationship between the observed candidate and all the other candidates that competes with the observed candidate in the same tableau. These inequalities are built based on the fact that the observed candidate should always have a higher Harmony than any of its competitors. As with the learned weights by the MaxEnt learner, each of the weights is constrainted to be non-negative in the LP problem. Since there is potentially an infinite number of solutions that satisfy such inequality relations, the objective function for the LP problem is defined to find the set of weights such that the sum of weights is minimized. A Linear Programming problem solver is a mathematical algorithm that finds finds such a set of weights. See Potts et al. (2010) for a more in-depth description of how the LP problem is set up.

The existing software implementation of Potts et al. (2010)'s LP solver, OT-Help (Staubs et al., 2010), was not sufficient to solve the patterns that we investigate for two reasons. First, OT-Help was not sufficient to handle the large number of tableaux and candidates in the learning problems that we deal with in this paper. Second, OT-Help was not built to find solutions when there are multiple hidden representations that are consistent with one overt surface representation. The only way OT-Help could check if there exists a solution for a given stress pattern was to generate the factorial typology, and then check if the pattern in question exists in the typology. In our new implementation, SoftStress (Lee & Pater, 2024), we deal with both of these problems, which we discuss next. SoftStress is implemented in Python, using the 'PuLP' package (Mitchell et al., 2011) and uses PuLP's default solver 'Cbc (COIN-OR Branch and Cut)' (Forrest, 2002), which are commonly used tools to set up Linear Programming problems.

2.2.1 *Candidates in tableaux* We define our learning problem first, to illustrate the magnitude of the learning problem we investigate.

Following Tesar & Smolensky (2000), we define a single stress pattern to be a set of SRs that has all combinations of heavy and light syllables for strings of 2-5 syllables in length, and for strings of light syllables 6-7 syllables in length. This means that we need to set up 62 tableaux in total for each stress system (see Table 7). In each of these tableaux, the SR candidates are all the possible ways that primary and secondary stresses can be assigned to that string of syllables. The SR candidates are included in the tableaux, based on the assumption that each word always has one main stress and can have any number of secondary stresses. We then include all possible HR candidates that are compatible with each SR, where HRs are all possible prosodifications, assuming feet can either be monosyllabic, or disyllabic, and right-headed or left-headed.

For example, for two-syllable-long words, we set up one tableau for each of the four possible combinations of heavy and light syllables as in (4): [L L], [L H], [H L] and [H H]. In each of these tableaux, there are four possible SR candidates, as shown in the second row in (4) for the tableau for [L L]. 1 indicates primary stress and 2 indicatees secondary stress. All possible HRs for each of these four SRs are given next. The HRs that are compatible are grouped with large square brackets, e.g., [(L L1)] and [L (L1)] are both consistent with the SR [L L1].

(4) Two-syllable-long words

- 4 URs: [L L], [L H], [H L], [H H]
- 4 SRs in the tableau for [L L]: [L L1], [L1 L], [L1 L2], [L2, L1]
- 6 HRs for [L L]: $\left[\text{[(L L1)], [L (L1)]}\right]$, $\left[\text{[(L1 L)], [(L1) L]}\right]$, $\left[\text{[(L1) (L2)]}\right]$, $\left[\text{[(L2) (L1)]}\right]$

To find the weights that make the observed SR optimal, we set up the LP problem containing the inequalities. For each tableau, we need to set up 'n-1' inequalities where 'n' equals the number of candidates

in the tableau, because the optimal candidate needs to be compared with each of the other candidates. In our learning problem, the number of candidates, and thus the number of equations, grows fast with the increase of word length, as presented in (7). Assuming we do not have the HRs in our learning data, we would still need to set up 3762 inequalities (3824 SRs - 62 optimal candidates) where the number of variables in each of these inequalities is the number of constraints.

SR?: Overt form or full structure (say what we mean it's not phonological surface representation that includes hidden structure)

Length	URs/length	SRs/UR	HRs/UR
2	4	4	6
3	8	12	24
4	16	32	88
5	32	80	300
6	1	192	984
7	1	448	3136
Total	URs: 62	SRs: 3824	HRs: 15344

Table 7: Number of URs for each word length, and the number of SR/HR candidates in each tableau (UR)

To avoid making our LP problem unnecessarily complicated, we add tableaux one by one to the LP problem, thereby checking whether there exists an HG solution for a subset of the stress pattern we want to solve. For instance, a quantity sensitive system that shifts the main stress according to the weight of the syllable is logically not representable if there is no constraint such as WEIGHT-TO-STRESS PRINCIPLE (WSP) in the constraint set, which requires heavy syllables to be stressed. In such a case, we only need to construct a simpler LP problem containing inequalities for two tableaux: [L L] and [H L], where the optimal SRs are [L L1] and [H1 L], for instance. We only need to construct the LP problem for these two tableaux to conclude there exists no HG solution for the given pattern with the current set of constraints that lacks WSP. The current implementation of LP stops adding more tableau once it fails to find a solution and informs the user that the current constraint set cannot represent the pattern.

2.2.2 *Finding multiple compatible solutions due to hidden structure* In a single LP problem, a series of inequalities are set up to compare one single candidate against each of the other competing candidates in the tableau. When there are multiple HR candidates that are consistent with a single observed SR, as we do in a foot-based grammar, we need to check all combinations of HR candidates consistent with the observed SR forms across tableaux.

To illustrate this point, consider a seemingly simple stress pattern we call 'Québecois'. It is a quantity-insensitive pattern, where main stress always falls on the final syllable, and a single secondary stress falls on the initial syllable. Therefore, in this pattern, the observed SRs are: [L2 L1], [L2 H1], [H2 L1], [H2 H1], [L2 L L1], [L2 L H1], for example. Québecois is representable and learnable using a foot-based constraint set that has constraints that require feet to be left-aligned trochee (ALLFEET-L, TROCHEE) and one that require the main stress to be on the right edge (MAIN-SYL-R).

However, once we set up the LP problem for Québecois and check all combinations of the HR candidates that are consistent with the observed SRs, it turns out that there are fifteen distinct solutions that make different HR candidates optimal. The solutions uniformly use constraints that are critically needed (TROCHEE, MAIN-SYL-R, ALLFEET-L) but vary in using other constraints. Table 8 shows 4 out of 15 solutions found for this pattern using the constraint set introduced in §3.2.1. Each numbered solution is a vector of weights for constraints listed in the leftmost column. We omit constraints with zero weights from the table. On the bottom of Table 8, the optimal HR given the constraint weights are listed. It shows that the constraints make distinct HRs optimal, depending on their weights, but crucially all these solutions result in making the observed SR optimal.

It is important to note that though there are multiple possible solutions that can be found by the LP solver, the MaxEnt hidden structure learner deterministically learns only one of them. The rightmost column shows the weights learned by the MaxEnt hidden structure learner, when the weights have been initialized at 1, which make the same HR candidates most optimal in terms of Harmony, as the solution in (3) does. If the learner is initialized with different weights, it may learn a different solution. We have not investigated

	Solutions found with constraints in 6				
Constraints	(1)	(2)	(3)	(4)	Learned (=(3))
TROCHEE	5	12	4	2	9.124
MAIN-SYL-R	4	5	5	1	12.283
ALLFEET-L	4	6	3	1	5.163
PARSE	3	8	2	0	4.208
IAMB	2	8	0	1	2.168
WORDFOOT-R	1	1	0	1	3.315
WORDFOOT-L	0	0	0	1	5.878
ALLFEET-R	0	1	0	0	1.744
FTBIN	0	2	0	0	0
WSP	0	0	0	0	0.439
...	0	0	0	0	0
Observed SR	Optimal HRs by each solution				
[L2 L L1]	[(L2) (L L1)]	[(L2) (L L1)]	[(L2 L) (L1)]	[(L2) L (L1)]	[(L2 L) (L1)]
[L2 L H1]	[(L2) (L H1)]	[(L2 L) (H1)]	[(L2 L) (H1)]	[(L2) L (H1)]	[(L2 L) (H1)]
...	...	...	...	...	...

Table 8: 4 distinct solutions among 15 possible solutions for Québecois, and the learned constraint weights (when constraint weights were initialized at 1) which make the same HR candidates optimal as Solution (3) does. Only constraints with non-zero weights are shown.

whether some solutions are 'easier' to learn, in the sense that when the learner is initialized with random weights, it is more likely to learn one particular solution than another. We leave for future work to investigate such questions.

Solving of a pattern can take a very long time when there are many solutions, as the LP solver needs to check all possible combinations of HR candidates consistent with the observed SRs. In testing the representability of the stress patterns we investigate, we therefore take the approach of trying to learn the pattern first and then checking the representability later. We first run the MaxEnt hidden structure learner with a fixed number of learning epochs (10,000), and a learning rate of 4, using the Gradient Descent optimization algorithm. Constraint weights are initialized at 1. If the learner fails to learn the pattern within 10,000 epochs, which we define to be reaching the summed probability over 0.90 for the observed SR, we only then try to check if the failed patterns are in fact representable, using our LP solver.

There are three possible outcomes following these two steps (learning and solving): a learning success, a learning failure followed by a solving success, and a learning failure followed by a solving failure. If learning succeeds, then it suggests that there exists at least one solution for the pattern, and the current constraint set can represent the pattern. In this case, we do not bother running the solver on the pattern. On the contrary, if learning fails despite there being a solution, it suggests that the learner is trapped in a local optimum along the process of learning as we've shown in §2.1.1. Finally, if both learning and solving fail, then the pattern is not representable by the current constraint set. As tableaux are added to the LP problem one by one (§2.2.1), it often does not take a long time for the solver to realize the pattern is not representable.

3 Case study: Testing the learnability and representability of attested stress patterns

With the MaxEnt hidden structure learner and the LP solver described in §2, we present a case study of how theories of grammar can be compared in terms of their learnability of attested stress patterns.

3.1 *Target languages* We investigate 61 attested stress patterns documented in the database 'StressTyp2' (Goedemans et al., 2015). These patterns have been selected as they are the patterns that are represented as finite state transducers (FSTs). Among the patterns stored in StressTyp2, wwe have also excluded patterns that cannot be represented with the set of constraints we investigate, e.g., patterns that have

lexical exceptionality, ternary weight (superheavy, heavy, and light) systems, and so on. Each of these patterns is represented with three digits, which can be entered in the following URL in place of 'XXX' to access additional information about the pattern in StressTyp2: `http://st2.ullet.net/?content=browse_pattern&pattern=XXX`.

3.2 *Constraint sets* We compare two grammatical theories that differ in how they represent metrical structure: foot-based theories and grid-based theories. The difference between these theories is represented as the set of candidates and constraints in Tableaux used for the learning. We investigate two foot-based constraint sets, and two grid-based constraint sets.

3.2.1 *Foot-based constraints* For foot-based theories, we adopt the foot-based constraint set based on the proposals in Prince & Smolensky (1993/2004) and McCarthy & Prince (1993), which we call 'TesarSmolensky(TS)-Original'. Definitions of constraints in this set are in (5). Note that definitions of constraints that target either the left or the right edge are merged below (e.g., WORDFOOT-R and WORDFOOT-L are merged as WORDFOOT-R/L).

(5) TS-Original constraints (n=12)

- NONFIN: Assign a violation if the final syllable of a word is footed.
- FTBIN: Assign a violation for each foot that consists only of a light syllable.
- IAMB: Assign a violation for each foot if its final syllable is not stressed (no violation for monosyllabic feet).
- FOOTNONFIN: Assign a violation for each foot if its final syllable is stressed (every monosyllabic foot gets one violation).
- PARSE: Assign a violation for each syllable that is not footed.
- WSP: Assign a violation for every heavy syllable that is not stressed.
- WORDFOOT-R/L: Assign a violation if the final/initial syllable of a word is not footed.
- MAIN-R/L: Assign a violation for every syllable between the right/left edge of the word and the right/left edge of the foot that has primary stress.
- ALLFEET-R/L: Assign a violation for every syllable between the right/left edge of the word and the right/left edge of each foot.

We created a revised TS constraint set ('TS-revised') by making changes to some of the constraints in (5) and adding NONFIN-MAIN constraint. The revisions and definitions of revised constraints are in (6). Instead of FOOTNONFIN, which assigns a violation for every monosyllabic feet, we included TROCHEE which does not. We also revised the main stress assigning constraints to count the number of intervening syllables between word edges and the main stressed syllable instead of the main stressed foot edge.

(6) TS-Revised constraints (n=13): 3 revisions + NONFIN-MAIN

- TROCHEE (instead of FOOTNONFIN): Assign a violation for each foot if its initial syllable is not stressed (no violation for monosyllabic feet).
- MAIN-SYL-R/L (instead of MAIN-R/L): Assign a violation for every syllable between the right/left edge of the word and the syllable that bears main stress.
- NONFIN-MAIN: Assign a violation to a final syllable that has primary stress.

3.2.2 *Grid-based constraints* The grid-based representation of metrical structure is proposed in Gordon (2002), such that the representation does not have the structural ambiguity with respect to the learning problems that we have shown for the foot-based representation. Therefore, the MaxEnt learner is guaranteed to find a correct grammar as long as the given constraints can represent the pattern.

We adopt the constraint set used in Gordon (2002), which we call 'Gordon-Original'. As the original grid-based constraint set in Gordon is proposed to capture a typology of attested quantity-insensitive stress sysems, we supplemented an additional WEIGHT-TO-STRESS PRINCIPLE (WSP) constraint to additionally

capture the quantity-sensitive systems. Constraints are listed in (7). We rephrased the constraint definition of WSP in terms of grid marks. For the definitions of these grid-based constraints that are rephrased using the 'Assign a violation for every...' format, see Example (2) in Lee et al. (2023:3).

(7) Gordon-Original constraints (n=13): constraints from Gordon (2002), plus WSP

- ALIGN(X_1, L, 0, PrWd), ALIGN(X_1, R, 0, PrWd), ALIGN(EDGES, level 0, PrWd, X_1) (i.e., ALIGNEDGES), NONFINALITY, *CLASH, *LAPSE, *EXTENDEDLAPSE, *LAPSERIGHT, *LAPSELEFT, *EXTENDEDLAPSERIGHT, ALIGN(X_2, L, 1, PrWd), ALIGN(X_2, R, 1, PrWd)
- WSP: Assign a violation for every heavy syllable that does not have a level 1 grid mark

We also examine a constraint set that revises the main stress placement constraints to count the number of syllables from the edge, parallel to the revised foot-based main stress constraints. The revised main stress constraints are in (8). We refer to this constraint set with the revised main stress constraints as 'Gordon-revised'.

(8) Gordon-Revised constraints (n=13): 2 revisions

- ALIGN(X_2, L, 0, PrWd) (instead of ALIGN(X_2, L, 1, PrWd)): Assign a violation for every grid mark at level 0 intervening between each grid mark at level 2 and the left edge of the word.
- ALIGN(X_2, R, 0, PrWd) (instead of ALIGN(X_2, R, 1, PrWd)): Assign a violation for every grid mark at level 0 intervening between each grid mark at level 2 and the right edge of the word.

4 Results

We descriptively report our major results in this section. As mentioned in the introduction, for a more thorough discussion of our results, see Lee et al. (2024).

Constraint set	QI (28)	QS (33)	All (61)
TS-Original	19/28 (67.9%)	12/33 (36.4%)	31/61 (50.8%)
TS-Revised	23/28 (82.1%)	20/33 (60.6%)	43/61 (70.5%)
Gordon-Original/Revised	28/28 (100%)	13/33 (39.4%)	41/61 (67.2%)

Table 9: Representational capacity of the constraint sets in §3.2 for 61 attested stress patterns

Table 9 summarizes how constraint sets fare in terms of representing quantity-insensitive (QI) and quantity-sensitive (QS) patterns. Gordon-Original and Gordon-Revised turned out to represent the same patterns and therefore presented in the same row. Making the revisions in (6) resulted in an improvement of representing 4 more quantity insensitive (Québecois being one of them) and 8 more quantity sensitive patterns. While grid-based constraints captured all of the QI patterns, simply adding WSP to the constraint set only made it capture 13/33 QS patterns. On the contrary, while foot-based constraints represented more QS patterns than grid-based constraints did, they were still not able to represent 5/28 QI patterns. There were also 13 quantity sensitive patterns that are not representable by any of the constraint sets.

Constraint Set	QI	QS	All
TS-Original	0.95 (18/19)	0.83 (10/12)	0.90 (28/31)
TS-Revised	0.83 (19/23)	0.95 (19/20)	0.88 (38/43)

Table 10: Rate of learning success, with the weight initialization at 1, for the patterns that are representable with foot-based constraint sets

Next, we report success rates for the foot-based constraint sets in Table 10, when constraint weights are initialized at 1. For the TS-Original constraint set, two QS patterns fail to be learned (156 Fijian and 200 Nyawaygi), and in addition, one of the QI patterns is not learned (212 Southern Paiute). For TS-Revised, four QI patterns fail to be learned: 157 Garawa, 117 Lenakel, 158 Georgian and 170 Indonesian, as well as

QS Mongolian Khalka. In general, QI patterns are learned better, in terms of success rate, with TS-Original constraints whereas QS patterns are learned better with TS-Revised constraints.

A noteworthy pattern we observe from Table 10 is that an improved representational capacity does not automatically imply an improvement in learnability. The revisions caused four more QI patterns to be represented, but also resulted in a lower learning success rate. Also, it was possible that a QI language that is both representable and learnable (such as 157 Garawa) before the revision (TS-Original) may not be learnable with the constraint revisions (TS-Revised) despite being representable. Lee et al. (2024) discusses why the learner fails to learn the Garawa pattern with TS-Revised constraints. On the other hand, the revision resulted in improvements in both representability and learnability for QS patterns.

5 Discussion and future directions

In this paper, we have pursued what seems to be a novel approach to the study of learning in generative grammar. Prior work studies the learning, with a given theory of grammar (see Dresher & Kaye, 1990 and Gibson & Wexler, 1994 for pioneering work in this tradition). Here we take a given learning theory, and study the consequences of altering the grammatical theory.

We have introduced new software that provides computational tools that find the constraint weights in HG/MaxEnt tableaux when there is hidden structure in the learning data. In particular, we made an improvement on an existing software implementation that finds HG solutions (Potts et al., 2010; Staubs et al., 2010), so that all possible distinct but compatible solutions can be found for a single pattern.

We presented a initial set of results on the representation and learning of a set of attested stress patterns from StressTyp2 (Goedemans et al., 2015). We showed that the original TS constraint set of Tesar & Smolensky (2000:*TS*), which we refer to as TS-Original, captures a subset of the patterns that a slightly revised foot-based constraint set does (TS-Revised), and also a subset of the patterns that can be generated with a set of grid-based constraints based on Gordon (2002). In terms of learning, we identified that the TS-Original constraints are at an advantage over TS-Revised for quantity insensitive patterns, in general, while the opposite is true for the quantity sensitive patterns.

We conclude the paper with three future directions.

First, all of the constraint sets that we studied here cannot represent the full attested typology. As we briefly noted, there are also patterns that we left out in our investigation, such as ones involving ternary distinctions in weight. The development of alternative constraint sets for stress, and the computational assessment of their representational capacity and learnability, is one direction for future work.

Next, the learning problems studied here are highly idealized in the sense that we assume words are sequences of light and heavy syllables, though the weight contrast itself is potentially something the learners need to figure out. We also make simplifying assumptions that there is no lexically specific exceptions in the learning data, and that words of different length appear with the same probability in the learning data. Dropping some of these assumptions is another clear future direction for future studies.

Finally, the same approach to learning and typology we have taken here could be extended to other linguistic domains, including ones outside of phonology, like syntax (see e.g. (Prickett et al., 2019) on MaxEnt learning of syntax). The software that we have used for this work is freely available, which we hope will aid in pursuing this research.

References

Anttila, Arto & Giorgio Magri (2018). Does MaxEnt Overgenerate? Implicational Universals in Maximum Entropy Grammar. *Proceedings of the Annual Meetings on Phonology* 5, URL `http://dx.doi.org/10.3765/amp.v5i0.4260`.

Boersma, Paul & Joe Pater (2016). Convergence properties of a gradual learner in Harmonic Grammar. McCarthy, John J. & Joe Pater (eds.), *Harmonic Grammar and Harmonic Serialism*, Equinox Press, Bristol, Connecticut, 389–434.

Dresher, B. Elan & Jonathan D. Kaye (1990). A computational learning model for metrical phonology. *Cognition* 34:2, 137–195, URL `https://linkinghub.elsevier.com/retrieve/pii/001002779090042I`.

Forrest, John (2002). Cbc (Coin-or branch and cut). URL `https://github.com/coin-or/Cbc`.

Gibson, Edward & Kenneth Wexler (1994). Triggers. *Linguistic Inquiry* 25:3, 407–454, URL http://www.jstor.org/stable/4178869.

Goedemans, Rob, Jeffrey Heinz & Harry Van der Hulst (2015). StressTyp2, version 1. Web download archive. http://st2.ullet.net/.

Goldwater, Sharon & Mark Johnson (2003). Learning ot constraint rankings using a maximum entropy model. *Proceedings of the Stockholm workshop on variation within Optimality Theory*, vol. 111, p. 120.

Gordon, Matthew (2002). A factorial typology of quantity-insensitive stress. *Natural Language & Linguistic Theory* 20:3, 491–552.

Gordon, Matthew (2011). Stress systems. *The Handbook of Phonological Theory*, Wiley, p. 141–163, URL http://dx.doi.org/10.1002/9781444343069.ch5.

Jarosz, Gaja (2013). Learning with hidden structure in optimality theory and harmonic grammar: Beyond robust interpretive parsing. *Phonology* 30:1, 27–71.

Jarosz, Gaja (2015). Expectation driven learning of phonology. *Ms. University of Massachusetts Amherst* .

Jarosz, Gaja (2019). Computational modeling of phonological learning. *Annual Review of Linguistics* 5:1, p. 67–90, URL http://dx.doi.org/10.1146/annurev-linguistics-011718-011832.

Lee, Seung Suk & Joe Pater (2024). SoftStress software. https://github.com/seungsuklee/SoftStress.

Lee, Seung Suk, Alessa Farinella, Cerys Hughes & Joe Pater (2023). Learning stress with feet and grids. *Proceedings of the Annual Meetings on Phonology*, vol. 10.

Lee, Seung Suk, Brandon Prickett & Joe Pater (2024). Representing and Learning Stress: A MaxEnt Framework for Comparing Learning Across Grammatical Theories . *Ms, University of Massachusetts Amherst* URL https://lingbuzz.net/008651.

McCarthy, John J & Alan Prince (1993). Generalized alignment. *Yearbook of Morphology 1993*, Springer, 79–153.

Mitchell, Stuart, Michael OSullivan & Iain Dunning (2011). Pulp: a linear programming toolkit for Python. *The University of Auckland, Auckland, New Zealand* 65, 1–12.

Pater, Joe (2016). Universal Grammar with Weighted Constraints. McCarthy, John J. & Joe Pater (eds.), *Harmonic Grammar and Harmonic Serialism*, Equinox Publishing, Bristol, Connecticut, 1–46.

Pater, Joe & Brandon Prickett (2022). Typological gaps in iambic nonfinality correlate with learning difficulty. *Proceedings of the Annual Meetings on Phonology*, vol. 9.

Pater, Joe, Robert Staubs, Karen Jesney & Brian Cantwell Smith (2012). Learning probabilities over underlying representations. *Proceedings of the twelfth meeting of the Special Interest Group on Computational Morphology and Phonology*, 62–71.

Potts, Christopher, Joe Pater, Karen Jesney, Rajesh Bhatt & Michael Becker (2010). Harmonic grammar with linear programming: from linear systems to linguistic typology. *Phonology* 27:1, 77–117.

Prickett, Brandon & Joe Pater (2024). Hidden Structure MaxEnt Learner. https://github.com/blprickett/Hidden-Structure-MaxEnt.

Prickett, Brandon, Kaden Holladay, Shay Hucklebridge, Max Nelson, Rajesh Bhatt, Gaja Jarosz, Kyle Johnson, Aleksei Nazarov & Joe Pater (2019). Learning syntactic parameter settings without triggers by assigning credit and blame. *Proceedings from the Annual Meeting of the Chicago Linguistic Society* 55:1, 337–350, URL https://www.ingentaconnect.com/content/cls/pcls/2019/00000055/00000001/art00025.

Prince, Alan & Paul Smolensky (1993/2004). *Optimality Theory: Constraint interaction in generative grammar*. Blackwell.

Smolensky, Paul & Géraldine Legendre (2006). *The harmonic mind: From neural computation to optimality-theoretic grammar (Linguistic and philosophical implications)*. MIT Press.

Staubs, Robert, Christopher Potts, Patrick Pratt, John McCarthy & Joe Pater (2010). OT-Help 2.0.

Tesar, Bruce & Paul Smolensky (2000). *Learnability in Optimality Theory*. MIT Press.