Modeling island effects with probabilistic tier-based strictly local grammars over trees

Charles Torres
University of California, Irvine
charlt4@uci.edu

Kenneth Hanson
Stony Brook University
kenneth.hanson@stonybrook.edu

Thomas Graf
Stony Brook University
mail@thomasgraf.net

Connor Mayer
University of California, Irvine
cjmayer@uci.edu

Abstract

We fuse two recent strands of work in subregular linguistics—probabilistic tier projections (Mayer, 2021) and tier-based perspectives on movement (Graf, 2022a)—into a probabilistic model of syntax that makes it easy to add gradience to traditional, categorical analyses from the syntactic literature. As a case study, we test this model on experimental data from Sprouse et al. (2016) for a number of island effects in English. We show that the model correctly replicates the superadditive effects and gradience that have been observed in the psycholinguistic literature.

1 Introduction

Gradience has been a long-standing issue in theoretical syntax and its interface with psycholinguistics. Is gradience a performance phenomenon or part of syntax proper? And if the latter, how could current syntactic formalisms handle gradience considering they were designed around the categorical distinction between well-formed and ill-formed structures? In this paper, we approach the issue of gradience from the perspective of *subregular linguistics*, a program equally rooted in theoretical linguistics and formal language theory. Subregular linguistics seeks to identify very restricted classes of computational (string or tree) mechanisms that can capture a wide range of linguistic phenomena. The insights from this perspective can be leveraged in a variety of ways, e.g. for new learning algorithms, novel explanations of typological gaps or linguistic universals, or to identify abstract properties that hold of both phonology and syntax.

We combine recent subregular work by Graf (2018, 2022b,a) on syntactic movement as a *tier-based strictly local* (TSL) dependency over trees with the framework in Mayer (2021) for probabilistic TSL dependencies over strings. Intuitively, a dependency is TSL iff it can be analyzed in two steps: first, one projects a tier that contains only some parts of the original structure, and second, this tier must satisfy a finite number of well-formedness constraints on adjacent structural elements. Mayer's framework allows for gradience in the string case of TSL by making this tier projection probabilistic while keeping the constraints categorical. We extend this notion of probabilistic tier projection to the the kind of TSL over trees that is used by Graf to capture syntactic movement.

The resulting framework of probabilistic TSL dependencies over trees can account for key aspects of the gradient judgments commonly observed with *island effects*, where a phrase is illicitly moved out of a containing phrase that does not allow for extraction. An example of such an island violation is shown below.

(1) a. Who does Mary say that John likes? (no island)

b. ?? Who does Mary wonder whether John likes? (*whether* island)

Concretely, we test the ability of a probabilistic TSL model to handle a subset of the experimental island data in Sprouse et al. (2016).[1] The gradience observed in this experimental island data is arguably the result of many interacting factors, which may also include performance, semantics, and pragmatics (see Chaves (2022) for a recent survey). We conclude that if one wants to capture the syntactic aspects of said gradience directly in the grammar, it is eminently feasible to do so — the switch from categorical to gradient is computationally simple, natural, and does not require any modifications of the underlying syntactic analysis.

Our paper makes several contributions beyond showing the empirical viability of probabilistic TSL over trees. It continues a recent trend in subregular linguistics to increasingly unify phonology and syntax, with both aspects of language using

[1] We thank Jon Sprouse for giving us permission to use the experimental data for English from Sprouse et al. (2016).

roughly the same kind of dependencies but applying them over strings and trees, respectively. In doing so, it also lends additional support to the specific proposals about movement in Graf (2018, 2022b,a) and gradience in Mayer (2021). The view of movement as a TSL dependency is not a mere stipulation that works in the limited case of categorical judgments, but rather provides exactly the kind of parameters that are also needed for gradience. TSL thus seems to capture a fundamental aspect of movement. Similarly, the probabilistic tier projections of Mayer (2021) have broad empirical appeal that extends far beyond the phenomena that they were originally proposed for. At the same time, our paper responds to the challenge by Chaves and Putnam (2022) to provide a TSL model of syntax that can handle gradient data. The fact that this answer requires no major changes to the categorical analysis supports the position commonly espoused by syntacticians that the issue of gradience is largely orthogonal to the enterprise of identifying the relevant syntactic structures and the operations and constraints that give rise to them.

The paper proceeds as follows. The Background section (§2) covers the relevant subregular concepts over strings. It first introduces the categorical notion of TSL (§2.1) before generalizing it to probabilistic TSL (§2.2, 2.3). We then turn to TSL over trees (§3), starting with an intuitive introduction of movement as a TSL dependency over trees and how this can be used to capture island effects in a categorical setting (§3.1–3.3). This intuition is then spelled out in formal terms (§3.4) that make it easy to combine tree TSL with the probabilistic notion of TSL from §2.3. Finally, we present the results of a modeling study (§4) showing that a simple probabilistic TSL grammar can predict many of the salient properties of the experimental data on island effects from Sprouse et al. (2016). We close with a brief discussion of the results (§5).

2 Background

This section introduces all relevant mathematical aspects of the probabilistic TSL formalism. Throughout we let Σ be an alphabet of symbols, ε the empty string, Σ^* the Kleene closure of Σ (the set of all strings of length 0 or more formed over Σ), and Σ^k the largest subset of Σ^* that contains only strings of length k. The symbols $\rtimes$ and $\ltimes$ represent left and right string boundary symbols, respectively. The : operator has type $\Sigma \rightarrow (\Sigma^* \rightarrow \Sigma^*)$ and prepends a symbol in Σ to a string in Σ^* (e.g. $a{:}bc = abc$).

2.1 Strictly local and tier-based strictly local languages

Let $s \in \Sigma^*$ for some Σ. The set of k-factors of s, $f_k(s)$, is defined as all the substrings of $\rtimes^{k-1} s \ltimes^{k-1}$ of length k. For example, $f_2(\text{tree}) = \{\rtimes\text{t}, \text{tr}, \text{re}, \text{ee}, \text{e}\ltimes\}$.

A *strictly k-local* (SL-k) grammar is a set G that contains (finitely many) forbidden substrings of length k. A string s is well-formed with respect to G iff $f_k(s) \cap G = \emptyset$, i.e. if it contains no illicit substrings of length k.[2]

Heinz et al. (2011) define a *tier-based strictly k-local* (TSL-k) grammar as a tuple $\langle G, T\rangle$ such that $T \subseteq \Sigma$ is a *tier alphabet* and $G \subseteq T^k$ is a SL-k grammar over the tier alphabet. The tier projection function π_T, which deletes from any given string all symbols not in T, is defined recursively:

$$\pi_T(\varepsilon) := \varepsilon \tag{1}$$

$$\pi_T(\sigma u) := \begin{cases} \sigma\pi_T(u), & \text{if } \sigma \in T \\ \pi_T(u), & \text{otherwise} \end{cases} \tag{2}$$

where $\sigma \in \Sigma$ and $u \in \Sigma^*$. The shape of the tier $\pi_T(s)$ projected from string s is then constrained by G exactly as in an SL grammar. Hence a string s is well-formed with respect to a TSL-k grammar $\langle G, T\rangle$ iff $f_k(\pi_T(s)) \cap G = \emptyset$.

A stringset (or equivalently, string language) is SL (TSL) iff it contains all and only those strings that are well-formed with respect to some SL-k (TSL-k) grammar, where $k \geq 0$.

2.2 Probabilistic tier projection

Probabilistic TSL (pTSL) is a generalization of TSL where π_T is a discrete probabilistic function.

A *discrete probabilistic function* $f : X \rightarrow (Y \rightarrow [0,1])$ maps pairs of strings $x \in X$ and $y \in Y$ to probabilities. These probabilities are drawn from the conditional distribution $P(y|x)$, and accordingly $\sum_{y \in Y} f(x,y) = 1$ for every $x \in X$.

Here we generalize the projection function π_T to a probabilistic version $\pi_P : \Sigma^* \rightarrow (\Sigma^* \rightarrow [0,1])$.

[2] Alternatively, a SL-k grammar can be interpreted as a collection of all well-formed substrings instead of all ill-formed substrings. In that case, string s is well-formed with respect to G iff $f_k(s)$ is a subset of G. The two interpretations are equivalent in the sense that every SL-k grammar G of forbidden k-grams generates the same set of strings as the SL-k grammar $(\Sigma \cup \{\rtimes, \ltimes\})^k - G$ of allowed k-grams.

Thus $\pi_P(x)$ returns a probability distribution over projections of some $x \in \Sigma^*$, and $\pi_P(x, y)$ returns the probability associated with projecting some $x \in \Sigma^*$ to some $y \in \Sigma^*$. It follows that $\sum_{y \in \Sigma^*} \pi_P(x, y) = 1$ for every $x \in \Sigma^*$. π_T is a special case of π_P such that the probability distribution for all $x \in \Sigma^*$ assigns a probability of 1 to a single projection.

The probabilistic tier projection π_P is calculated based on probabilities associated with the projection of each individual symbol in Σ. We define an additional function $P : \Sigma \to [0, 1]$. This function represents the probability that each symbol in Σ is projected to the tier. For example, if $P(a) = 0.7$, then there's a 70% chance the symbol a will project. We can then define π_P recursively as follows:

$$\pi_P(\varepsilon, v) := \begin{cases} 1, & \text{if } v = \varepsilon \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

$$\pi_P(\sigma_x{:}u, \varepsilon) := (1 - P(\sigma_x)) \cdot \pi_P(u, \varepsilon) \tag{4}$$

$$\begin{aligned} \pi_P(\sigma_x{:}u, \sigma_y{:}v) := \llbracket \sigma_x = \sigma_y \rrbracket \cdot P(\sigma_x) \cdot \pi_P(u, v) \\ + (1 - P(\sigma_x)) \cdot \pi_P(u, \sigma_y{:}v) \end{aligned} \tag{5}$$

where $\sigma_x, \sigma_y \in \Sigma$, $u, v \in \Sigma^*$ and $\llbracket \sigma_x = \sigma_y \rrbracket$ is an indicator function that evaluates to 1 if $\sigma_x = \sigma_y$ and 0 otherwise.

The base case (3) ensures that the only valid projection of ε is ε. In the first recursive case (4) where the input is non-empty and the projection is empty, the probability of the projection is the probability of not projecting each symbol in the input. In the second recursive case (5) where both input and projection are non-empty, we consider two possibilities for each symbol: either it projects (the first term), or it does not (the second term). The indicator variable ensures that we only consider projection as a possibility when the symbols at the beginning of the input and projection are identical.

Example. Let $\Sigma = \{a\}$ and $P(a) = 0.75$. We show that the probability of projecting $aa \to a$ is 0.375. First, by definition:

$$\begin{aligned} \pi_P(aa, a) = P(a) \cdot \pi_P(a, \varepsilon) \\ + (1 - P(a)) \cdot \pi_P(a, a) \end{aligned} \tag{6}$$

We omit the indicator variables for brevity. The first term corresponds to the case where the first a projects, and the second corresponds to the case where it does not. Solving for the two recursive instances of π_P in (6) gets us:

$$\begin{aligned} \pi_P(a, \varepsilon) &= (1 - P(a)) \cdot \pi_P(\varepsilon, \varepsilon) \\ &= 1 - P(a) \end{aligned} \tag{7}$$

$$\begin{aligned} \pi_P(a, a) &= P(a) \cdot \pi_P(\varepsilon, \varepsilon) \\ &\quad + (1 - P(a)) \cdot \pi_P(\varepsilon, a) \\ &= P(a) \end{aligned} \tag{8}$$

Plugging these into (6) gets us:

$$\begin{aligned} \pi_P(aa, a) &= P(a) \cdot (1 - P(a)) \\ &\quad + (1 - P(a)) \cdot P(a) \\ &= 0.75 \cdot 0.25 + 0.25 \cdot 0.75 \\ &= 0.375 \end{aligned} \tag{9}$$

The support of the distribution over projections, i.e. the set of projections assigned non-zero probability, is:

$$\begin{aligned} \pi_P(aa, aa) &= 0.5625 \\ \pi_P(aa, a) &= 0.375 \\ \pi_P(aa, \varepsilon) &= 0.0625 \end{aligned} \tag{10}$$

2.3 pTSL grammars

A pTSL-k grammar over an alphabet Σ is a tuple (π_P, G), where I) π_P is a probabilistic tier projection defined according to projection probabilities for each $\sigma \in \Sigma$, and II) $G \subseteq (\Sigma \cup \{\rtimes, \ltimes\})^k$ is a SL-k grammar.

The function $val_{(\pi_P, G)}$ defines the probability assigned to a string u by the grammar (π_P, G):

$$val_{(\pi_P, G)}(u) = \sum_{v \in \Sigma^*} \llbracket f_k(v) \cap G = \emptyset \rrbracket \cdot \pi_P(u, v) \tag{11}$$

where $\llbracket f_k(v) \cap G = \emptyset \rrbracket$ is an indicator variable that evaluates to 0 if v contains any illicit k-factors and 1 otherwise. $val_{(\pi_P, G)}(u)$ is the sum of the probabilities of all projections of the string u that do not contain any prohibited k-factors. Note that $val_{(\pi_P, G)}$ is not a probability distribution over input strings, but rather the conditional probability of some grammatical projection given the input string.

Example. Assume the definitions of Σ and π_P from the previous example, and suppose we have a pTSL-2 grammar where $G = \{aa\}$. Then:

$$\begin{aligned} val_{(\pi_P, G)}(aa) &= \pi_P(aa, a) + \pi_P(aa, \varepsilon) \\ &= 0.4375 \end{aligned} \tag{12}$$

$\pi_P(aa, aa)$ is not included in this calculation because the projection aa contains the prohibited substring aa.

In sum, a pTSL-k grammar is the combination of a categorical SL-k grammar G with a probabilistic tier projection π_P. In contrast to the categorical tier projection π_T, π_P may project multiple tiers from any given string s. Each one of these tiers has a specific probability that is the product of the projection probabilities that resulted in this tier given s. We then sum the probabilities of all tiers projected from s that are well-formed with respect to G, yielding the conditional probability of some grammatical projection given the input s. With this understanding of how TSL over strings may be made probabilistic, we now turn to TSL over trees.

3 (p)TSL over trees

Graf (2018) generalizes TSL (more precisely the subclass TSL-2) from strings to trees. The intuition is exactly the same as in the string case: Given a tree t over alphabet Σ, we project all nodes with a label in the tier alphabet $T \subseteq \Sigma$ while preserving the ordering between those nodes in terms of dominance and precedence. SL constraints then regulate the shape of permissible tiers. A full definition of TSL-2 over trees can be found in Graf and Kostyszyn (2021), but for present purposes only the tier projection needs to be discussed in depth.

The ensuing discussion is motivated by empirical examples such as the one below, which is an instance of an *island effect*.

(2) ?? Who does Mary wonder whether John likes t?

This sentence is commonly considered degraded by native speakers of English, and syntacticians attribute this to *whether* creating an island for extraction. In the parlance of Minimalist syntax, the object *who* in the embedded clause *wh*-moves to Spec,CP of the matrix clause, but *wh*-movement is degraded out of *whether*-clauses.

Let us see, then, how this can be captured with TSL over trees using the analysis in Graf (2022a). We will first put in place feature-annotated dependency trees as a tree-based representation of the syntactic derivation (§3.1), from which we project specific tree tiers to regulate movement in a strictly local manner (§3.2). This in turn provides an easy way of modeling a wide range of island constraints as a categorical constraint against specific movement configurations (§3.3). These intuitive ideas are then made rigorous and, ultimately, probabilistic in §3.4.

3.1 Syntactic representations

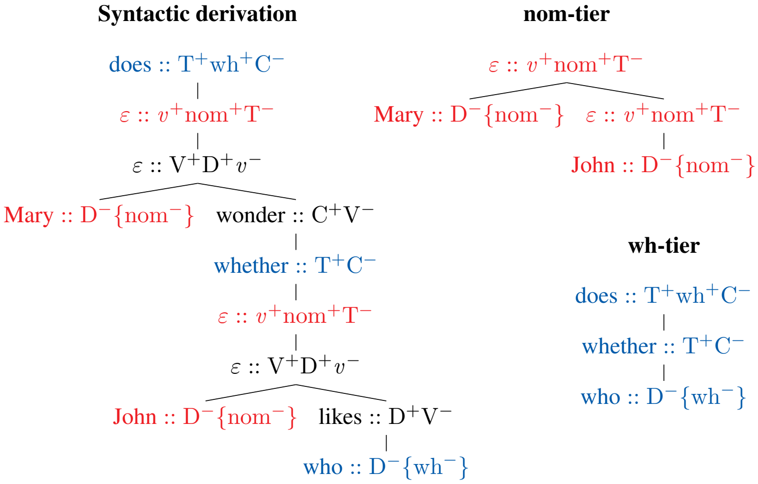
Each sentence is associated with a syntactic derivation, which we represent with a dependency tree. Figure 1 gives the dependency tree for (2). Following common Minimalist assumptions, each clause consists of a verb and its three extended projections: v (which selects the subject), T (which provides the default surface position for the subject), and C (which hosts complementizers and serves as a landing site for some movement steps). Each node of the dependency tree is a lexical item, and m is a mother of a iff m selects a as an argument. If a_1 and a_2 are both daughters of m, then a_1 is a right sibling of a_2 iff a_1 is selected by m before a_2 is. That is, the right-to-left order of siblings reflects the order of selection. The geometry of the dependency tree thus encodes all relevant head-argument relations and their relative order in the derivation.

In addition, every lexical item is given a *feature annotation* inspired by the feature system of Minimalist grammars (Stabler, 1997, 2011). For each lexical item, its feature annotation encodes its category (e.g. *category feature* X^-), the categories of its arguments (e.g. the string $\mathrm{X}^+\,\mathrm{Y}^+$ of *selector features*), whether it serves as a landing site for movement steps (e.g. *licensor feature* wh^+), and whether it undergoes any movement steps (e.g. the unordered set $\{\mathrm{nom}^-, \mathrm{wh}^-\}$ of *licensee features*).[3] Note the use of capitalization to distinguish category and selector features on the one hand from licensor and licensee features on the other. All four types of features will play a key role in deciding which nodes should be projected onto a given tier.

3.2 Movement tiers

With the basics of feature-annotated dependency trees in place, we turn to tier projection for movement. In Fig. 1, we have three separate movement steps: two instances of subject movement, and one instance of wh-movement. Let us consider the former first. The subject *Mary* in the matrix clause moves to Spec,TP of the matrix clause, and the subject *John* in the embedded clause moves to Spec,TP of the embedded clause. In both cases, this is implicitly encoded by the fact that *Mary* and *John* carry the licensee feature nom^-, and the

[3] In contrast to Minimalist grammars, licensee features are unordered in our system so that a mover with multiple licensee features will always target the closest dominating nodes with matching licensor features. This affects neither weak nor strong generative capacity (Graf et al., 2016) but is a crucial prerequisite for capturing movement dependencies via tiers.



Syntactic derivation
does :: $\mathrm{T}^+\mathrm{wh}^+\mathrm{C}^-$
ε :: $v^+\mathrm{nom}^+\mathrm{T}^-$
ε :: $\mathrm{V}^+\mathrm{D}^+v^-$
Mary :: $\mathrm{D}^-\{\mathrm{nom}^-\}$
wonder :: $\mathrm{C}^+\mathrm{V}^-$
whether :: $\mathrm{T}^+\mathrm{C}^-$
ε :: $v^+\mathrm{nom}^+\mathrm{T}^-$
ε :: $\mathrm{V}^+\mathrm{D}^+v^-$
John :: $\mathrm{D}^-\{\mathrm{nom}^-\}$
likes :: $\mathrm{D}^+\mathrm{V}^-$
who :: $\mathrm{D}^-\{\mathrm{wh}^-\}$
nom-tier
ε :: $v^+\mathrm{nom}^+\mathrm{T}^-$
Mary :: $\mathrm{D}^-\{\mathrm{nom}^-\}$
ε :: $v^+\mathrm{nom}^+\mathrm{T}^-$
John :: $\mathrm{D}^-\{\mathrm{nom}^-\}$
wh-tier
does :: $\mathrm{T}^+\mathrm{wh}^+\mathrm{C}^-$
whether :: $\mathrm{T}^+\mathrm{C}^-$
who :: $\mathrm{D}^-\{\mathrm{wh}^-\}$

Figure 1: Syntactic derivation for (2), its well-formed nom-tier (in red), and the ill-formed wh-tier (in blue)

corresponding T-heads carry the matching licensor feature nom^+. A lexical item with some licensee feature f^- will always move to a specifier of the closest dominating lexical item with matching licensor feature f^+. This is why *Mary* moves to Spec,TP of the matrix clause, whereas *John* moves to Spec,TP of the embedded clause. Each one of these subject movement steps is well-formed, and as pointed out by Graf (2018), this can be verified in a tier-based strictly local manner.

In order to determine whether the derivation contains any illicit instances of subject movement, we construct a subject movement tier that contains only nodes that matter for subject movement. At the very least, this tier must contain every lexical item that carries nom^+ or nom^- (as we will see during the discussion of wh-movement, projecting additional lexical items is exactly what gives rise to island effects). The resulting *nom-tier* is shown in Fig. 1. Note how the dominance relations in the tier match the dominance relations in the dependency tree. Moreover, *Mary* is the left sibling of the embedded T-head on the tier because in the dependency tree, *Mary* precedes the embedded T-head (that is to say, *Mary* is reflexively dominated by a node that is the left sibling of a node that reflexively dominates the embedded T-head). The nom-tier is well-formed iff it obeys both of the following conditions for movement tiers:

(3) **Well-formedness of an f-tier**

a. Every node with f^+ has exactly one node with f^- among its f-tier daughters.

b. Every node with f^- has an f-tier mother that carries f^+.

Both of these conditions are met in the nom-tier, which entails that all subject movement steps in the derivation are well-formed.

3.3 Categorical island effects

Now consider the case of wh-movement of *who* from the embedded object position to Spec,CP of the matrix clause. Without additional assumptions, this movement step should be well-formed. If we construct the corresponding wh-tier, it consists only of *does* with *who* as its only daughter. As the former carries wh^+ and the latter wh^-, the conditions in (3) are met and the tier should be well-formed. But we already saw that (2) is not considered well-formed due to the presence of *whether*. Suppose, then, that we also project *whether* onto the wh-tier, yielding the wh-tier in Fig. 1. Both conditions in (3) are now violated by the wh-tier because *whether* intervenes between *does* and *who*. Island effect thus arise whenever an element that does not carry the relevant features is projected onto a tier and destroys the mother-daughter configuration between a mover and its target.[4]

This same idea can be used to capture other island effects. In addition to the *whether island*

[4] Note that projecting *whether* on the nom-tier would not destroy any such configurations. Irrespective of whether one projects *whether*, the nom-tier is well-formed. In general, it is safe to assume that islands project onto all movement tiers, unless there is good empirical evidence that a specific movement type is not subject to a specific island condition. For the purposes of this paper, what exactly projects onto the nom-tier does not matter as all our modeling will focus exclusively on the wh-tier.

constraint described above, we will also examine the *adjunct island constraint* and the *complex NP constraint*. The adjunct island constraint prevents extraction from adjuncts, e.g. *because*-clauses as in (4a). The complex NP constraint prevents extraction from sentential complements of nouns (4b). Both effects also arise with extraction from relative clauses as in (4c) and (4d), respectively. For simplicity, we will conflate the difference between wh-movement and relative clause extraction and treat both as involving the features wh^+ and wh^- for the rest of this paper.

(4) a. * Who did Mary complain because John likes t?

b. * Who did Mary deny the rumor that John likes t?

c. * I saw the congressman who Mary worries if John respects t.

d. * I saw the man who Mary heard the rumor that John likes t.

All these cases can be analyzed as some lexical item projecting onto a movement tier and disrupting the local licensing relations there. The adjunct island constraints are captured by projecting the heads of adjunct islands, for example *because* and *if*. The complex NP constraint amounts to projecting all nouns that select a CP as their only argument (i.e. every lexical item whose feature annotation contains the substring $\mathrm{C}^+\mathrm{N}^-$). Crucially, the decision to project a lexical item only requires maximally local information: the surface realization of the lexical item and/or its feature annotation.[5]

However, all these accounts are hamstrung by the fact that tiers are either well-formed or ill-formed. It is not possible to express the fact that, say, *whether*-island violations are not judged as degraded as extraction from *because*-clauses. One easy way to add gradience to this system is to adapt the probabilistic tier projection mechanism of Mayer (2021), which we discussed in §2.2 and §2.3.

3.4 Probabilistic tree tier projection

In order to define a probabilistic tier projection for trees, we first need a rigorous definition of categorical tier projection for trees. We adopt the logic-based definition of Graf and Kostyszyn (2021) where a tier is just the result of enriching the dependency tree with relations for *tier daughter* and *tier sibling*.

Let us use $\triangleleft^+$ ($\triangleleft^*$) to denote proper (reflexive) dominance in the dependency tree, i.e. $x \triangleleft^+ y$ ($x \triangleleft^* y$) holds in dependency tree t iff x properly (reflexively) dominates y in t. We also use $x \prec y$ to denote that x is a left sibling of y in t. Furthermore, the predicate $T(x)$ is true iff the label of x (e.g. wonder :: $\mathrm{C}^+\mathrm{V}^-$ in Fig. 2) is part of our tier alphabet T. We define proper dominance on tier T ($\triangleleft_T^+$) and use that to subsequently define the daughter-of relation over tier T ($\triangleleft_T$), which in turn is needed to define the left-sibling relation over tier T ($\prec_T$):

$$\begin{aligned}
x \triangleleft_T^+ y &\Leftrightarrow T(x) \wedge T(y) \wedge x \triangleleft^+ y \\
x \triangleleft_T y &\Leftrightarrow x \triangleleft_T^+ y \wedge \neg\exists z[x \triangleleft_T^+ z \wedge z \triangleleft_T^+ y] \\
x \prec_T y &\Leftrightarrow \exists z[z \triangleleft_T x \wedge z \triangleleft_T y] \wedge \\
&\qquad \exists z, z'[z \triangleleft^* x \wedge z' \triangleleft^* y \wedge z \prec z']
\end{aligned}$$

These predicates implicitly define the tier T over dependency tree t and provide the relevant structural relations for tier constraints such as the one-to-one match between mothers with licensor features and daughters with licensee features we encountered in (3).

In order to turn this categorical notion of tree tiers into a probabilistic one, it suffices to make membership in the tier alphabet probabilistic. For example, if elements with the same label as x have a probability of 0.7 to project onto tier T, then the predicate $T(x)$ has a probability of 0.7 of being true. This is the only required change. The definitions of $\triangleleft_T^+$, $\triangleleft_T$, and $\prec_T$ remain exactly the same—it is only the interpretation of $T(x)$ that becomes probabilistic. Once this change is made, the probability of a given tier projection is calculated in exactly the same manner as in the string case (§2.2): it is the product of $T(x)$ for every x that projects, and $(1 - T(x))$ for every x that does not project. The overall conditional probability of a given tree having some grammatical projection is

[5] Mathematically, the tier projection may use any information that can be encoded in terms of a finitary annotation scheme for lexical items. This includes, among other things, the semantic denotation of the lexical item, a higher-dimensional vector representation derived from word embeddings, aspects of information structure such as topic and focus, or basic frequency information in terms of a finite classification system like *very rare*/*rare*/*common*/*ubiquitous*. Any kind of annotation that preserves Minimalist grammars' requirement that the set of lexical items must be finite is mathematically permissible. So even though we will limit ourselves to purely syntactic information in our subsequent discussion of island effects, the approach could be extended to consider at least some of the semantic and pragmatic factors observed in Chaves (2022) and the studies referenced therein.

also calculated in the same manner as the string case: it is the sum of the probabilities of all its possible licit tier projections.

4 Modeling study

The next section presents a computational modeling study where a simple pTSL grammar over trees is fit to experimental data on English island effects from Sprouse et al. (2016).[6] We demonstrate that in addition to exhibiting the superadditive effects found by Sprouse et al., it can also represent the gradience observed across judgments of different island effects.

4.1 Methods

The stimuli from Sprouse et al. (2016) were given a syntactic analysis using feature-annotated dependency trees as described in §3. We restricted ourselves to the subset of sentences exhibiting the island effects described above: *whether* islands, adjunct islands, and complex NP islands. We also omitted filler sentences. This produced a total of 160 trees.

Sprouse et al. (2016) partitions the data within each island effect type based on two factors: whether the sentence contains an island structure, and whether the node that undergoes movement is located in the matrix clause or the embedded clause. Examples of the four combinations of these two factors are show in (5) for *whether* islands (from Sprouse et al., 2016).

(5) a. Who t thinks [that John bought a car]? (non-island, matrix clause)

b. What do you think [that John bought t]? (non-island, embedded clause)

c. Who t wonders [whether John bought a car]? (island, matrix clause)

d. What do you wonder [whether John bought t]? (island, embedded)

This factorial design is intended to separate the effects of extracting from a matrix clause vs. extracting from an embedded clause, and also the effects of the presence or absence of an island structure. In particular, Sprouse et al. expect that sentences like (5d), which are the only ones that contain syntactic island configurations, should display *superadditive effects*. That is, the effect of these configurations on human judgments should be greater than the independent contributions of extracting out of an embedded clause, as in (5b), and the presence of an island structure that is not extracted over, as in (5c).

The dataset from Sprouse et al. (2016) contains about 14 Likert scale ratings for each sentence we considered. Because our model is unable to represent cross-speaker variability in judgments, we assigned each sentence the mean rating across participants. Following Sprouse et al. (2016), we use ratings that were Z-score normalized by participant rather than the raw Likert scores.

Using the dependency trees and Z-scores, we fit a pTSL grammar to the data by finding the optimal projection probabilities: that is, those that align as closely as possible the scores assigned by the model to the scores assigned by humans. We do this by first transforming the mean Z-score values to fall in the range $[0, 1]$ and then minimizing the mean squared error between the transformed human acceptability judgments and the probabilities assigned by the model. This minimization was performed using `scipy.optimize.minimize` (Virtanen et al., 2020) with bounded L-BFGS optimization to ensure each projection probability is within the interval $[0, 1]$.

We *a priori* fixed most of the projection probabilities to 0 (irrelevant nodes) or 1 (nodes with wh^+ or wh^-), and we fit only projection probabilities for nodes that could feasibly induce island effects. This was done to facilitate interpretability of the model, speed up the model training, and offset the comparatively small size of the training set.

The nodes whose projection probabilities were fitted were:

- that :: $\mathrm{T}^+\mathrm{C}^-$
- whether :: $\mathrm{T}^+\mathrm{C}^-$
- if :: $\mathrm{T}^+\mathrm{C}^-$
- all nodes whose feature annotation contains the substring $\mathrm{C}^+\mathrm{N}^-$

The first three items are potential blockers for the whether island and adjunct island constraints; nodes with $\mathrm{C}^+\mathrm{N}^-$ correspond to nouns that head complex NPs and should thus induce complex NP island effects. There are a set of seven nouns in the data that have this featurization (*rumor*, *claim*, etc.). For simplicity we assume all such nouns have the same projection probability and treat this as a single parameter.

Finally, nodes representing *wh*-movers and landing sites were set to always project. The latter

[6] The code and data can be found at: `https://github.com/connormayer/pTreeTSL`

includes interrogative C-heads and relative clause C-heads with feature string $\mathrm{T^+wh^+C^-}$ (recall that we use wh both for wh-movement and for relative clause movement). The former consists of wh-pronouns with the feature string $\mathrm{D^-\{wh^-\}}$.

Because fitting the model is stochastic, we performed training ten times in order to determine whether the probabilities reliably converge to the same values.

4.2 Results

For our four features of interest, learned projection probabilities showed little variance across the ten runs. Each converged within 10^{-3} to the same projection probability. This aligns with the suggestion in Mayer (2021) that the optimization function when fitting a pTSL model in this way is concave. We report projection probabilities and scores averaged across the ten runs.

Table 1 shows the projection probabilities learned by the model for the four nodes of interest. Recall that higher projection probabilities increase the likelihood of these nodes projecting to the wh-tier and intervening between a tier mother with $\mathrm{wh^+}$ and its tier daughter with $\mathrm{wh^-}$. Therefore, higher projection probabilities should correspond to lower ratings for the relevant island structures. The relative projection probabilities show that *if* is mostly likely to act as a blocker, *that* is least likely, and complex NPs and *whether* are intermediate between the other two.

The mean human scores and the mean model scores for each sentence type are shown in Fig. 2. The model scores capture several important aspects of the human judgments: (a) extracting out of a matrix clause is uniformly judged to be better than extracting out of an embedded clause; (b) extracting out of an embedded clause over an island produces the expected superadditive effects; and (c) the relative badness of the five types of island extraction (the right point in the red lines in Fig. 2) matches the relative badness reflected in the human judgments.

Node	Projection probability
that :: $\mathrm{T^+C^-}$	.46
$\mathrm{C^+\ N^-}$	.63
whether :: $\mathrm{T^+C^-}$	.73
if :: $\mathrm{T^+C^-}$	.89

Table 1: Mean projection probabilities

There are a number of aspects of the data the model fails to capture. First, it over-predicts this superadditivity in the case of relative clause adjunct islands, where it was not found in the human data. Second, it does a poor job of predicting the relative badness of forms in the matrix extraction condition. These sentences are not ungrammatical in terms of the wh-tier, and the model accordingly assigns them all probabilities of 1 in these cases. In particular, humans generally assign worse scores when an island structure is present, even if it is not a blocker, while the model cannot do so. Finally, although it captures the general tendency for extraction out of embedded clauses to be worse than extraction out of a matrix clause, the relative effect of this in different island types is not captured by the model.

5 Discussion

Assessing the performance of the probabilistic TSL model for the English island data from Sprouse et al. (2016) is a sutble affair because there are so many factors that could influence what Likert scores participants assign to specific stimuli. Syntactic constraints, processing difficulties, lexical frequency, semantics, pragmatics, and information structure may all be involved. By limiting our attention to only the phonetic exponents of lexical items and their feature make-up, we are asking the model to capture the experimental data as well as possible with only syntactic information. In that respect, the model succeeds as it gives rise to super-additivity, which has been argued to be the primary reflex of syntax in experimental island effect data.

Admittedly the model does not do a perfect job, and future work is needed to fully explore these issues. For example, the model overpredicts superadditivity in relative clause adjunct islands. This raises the question whether alternative analyses of relative clauses would have fared better in this respect, and if not, what non-syntactic factors could explain the less pronounced nature of superaddivity in these constructions. Similarly, although the model is able to capture superadditivity and the relative badness of the types of island violations considered here, it does poorly in predicting the variability in judgments of the non-island cases and the short forms of island cases. Once again this might indicate the need for a revised syntactic analysis, or point towards non-syntactic factors.

Crucially, these non-syntactic factors are not necessarily beyond the purview of the pTSL model —

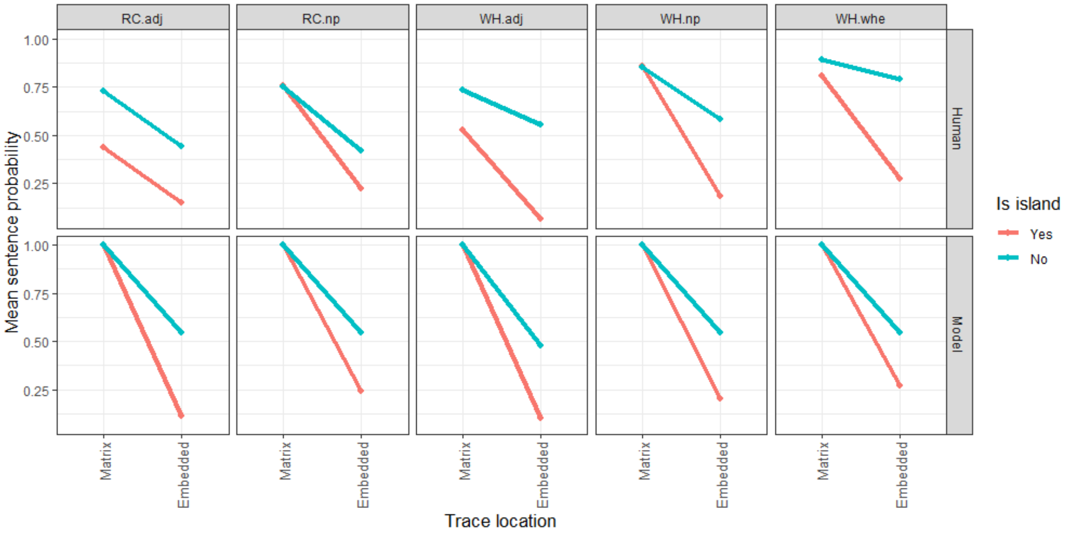


Figure 2: Human judgments from Sprouse et al. (2016) (top) and mean model judgments after training (bottom)

any information that can be lexicalized can be taken into account by the tier projection function. For example, an analysis that encodes topic and focus as movement of phrases to specific syntactic positions furnishes specific movement features that encode the topic-focus distinction and thus could serve as parameters for the tier projection and the constraints that apply on tiers. Hence it is important not to equate the syntax-only approach we took in this paper with the limits of what can be modeled with pTSL.

In relation to this, it is also important to remember that the probabilities themselves might encode remnants of non-syntactic factors and hence don't give us a "pure" picture of the role of syntax in island effects. It is likely that the learned projection probabilities shown in Table 1 encode some effects related to processing rather than syntax. In particular, *that* has a relatively high projection probability despite it not being considered a syntactic blocker. The model has likely assigned this probability in order to encode the decrease in acceptability between extraction out of a matrix clause and extraction out of an embedded clause. Integration of the model proposed here with other models, e.g. the processing approach of De Santo (2020) to gradience in adjunct islands, has the potential to shed more light on whether effects such as this should be modeled as part of the grammar.

Our primary goal was to show that the switch from categorical TSL (and the categorical syntactic analyses that can be expressed this way) to a probabilistic, gradient model is easy and empirically viable. The task of adequately modeling island effects with pTSL is much larger than this, but we are confident that pTSL will be able to provide novel insights in this domain.[7]

6 Conclusion

We have presented pTSL as a simple probabilistic extension of TSL syntax that makes it easy to add gradience to existing syntactic analyses (provided they can be stated in terms of categorical TSL). The key idea of this extension is the switch to a probabilistic tier projection function. We discussed island effects as an example of the empirical viability of this approach: the combination of a standard Minimalist analysis with probabilistic tier projection is able to replicate the superadditive effects of extraction out of islands and the gradience in the relative badness of different types of island constructions.

Acknowledgements

The work carried out by Kenneth Hanson and Thomas Graf for this project was supported by the National Science Foundation under Grant No. BCS-1845344.

[7] To avoid potential confusion: our pTSL model is not intended to be a model of how island effects are learned. The model is trained on scores assigned to the data in experimental contexts, which is not a realistic learning scenario. But the model is of interest for learning because its parameters are interpretable and it does successfully encode syntactic contributions to judgments of island effects, including superadditivity and gradience.

References

Rui P. Chaves. 2022. Sources of discreteness and gradience of island effects. *Languages*, 7:245.

Rui P. Chaves and Michael T. Putnam. 2022. Islands, expressiveness, and the theory/formalism confusion. *Theoretical Linguistics*, 48(3–4):219–231.

Aniello De Santo. 2020. MG parsing as a model of gradient acceptability in syntactic islands. In *Proceedings of the Society for Computation in Linguistics 2020*, pages 59–69. Association for Computational Linguistics.

Thomas Graf. 2018. Why movement comes for free once you have adjunction. *Proceedings of CLS*, 53:117–136.

Thomas Graf. 2022a. Subregular linguistics: Bridging theoretical linguistics and formal grammar. *Theoretical Linguistics*, 48:145–184.

Thomas Graf. 2022b. Typological implications of tier-based strictly local movement. In *Proceedings of the Society for Computation in Linguistics 2022*, pages 184–193, online. Association for Computational Linguistics.

Thomas Graf, Alëna Aksënova, and Aniello De Santo. 2016. A single movement normal form for Minimalist grammars. In *Formal Grammar: 20th and 21st International Conferences, FG 2015, Barcelona, Spain, August 2015, Revised Selected Papers. FG 2016, Bozen, Italy, August 2016*, pages 200–215, Berlin, Heidelberg. Springer.

Thomas Graf and Kalina Kostyszyn. 2021. Multiple wh-movement is not special: The subregular complexity of persistent features in Minimalist Grammars. In *Proceedings of the Society for Computation in Linguistics 2021*, pages 275–285, Online. Association for Computational Linguistics.

Jeffrey Heinz, Chetan Rawal, and Herbert G. Tanner. 2011. Tier-based strictly local constraints in phonology. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 58–64.

Connor Mayer. 2021. Capturing gradience in long-distance phonology using probabilistic tier-based strictly local grammars. In *Proceedings of the Society for Computation in Linguistics 2021*, pages 39–50, Online. Association for Computational Linguistics.

Jon Sprouse, Ivano Caponigro, Ciro Greco, and Carlo Cecchetto. 2016. Experimental syntax and the variation of island effects in English and Italian. *Natural Language & Linguistic Theory*, 34(1):307–344.

Edward P. Stabler. 1997. Derivational Minimalism. In Christian Retoré, editor, *Logical Aspects of Computational Linguistics*, volume 1328 of *Lecture Notes in Computer Science*, pages 68–95. Springer, Berlin.

Edward P. Stabler. 2011. Computational perspectives on Minimalism. In Cedric Boeckx, editor, *Oxford Handbook of Linguistic Minimalism*, pages 617–643. Oxford University Press, Oxford.

Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. 2020. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature Methods*, 17(3):261–272.