Sanskrit n-Retroflexion is Input-Output Tier-Based Strictly Local

Thomas Graf
Department of Linguistics
Stony Brook University
Stony Brook, NY 11794, USA
mail@thomasgraf.net

Connor Mayer
Department of Linguistics
University of California, Los Angeles
Los Angeles, CA 90046, USA
connormayer@ucla.edu

Abstract

Sanskrit /n/-retroflexion is one of the most complex segmental processes in phonology. While it is still star-free, it does not fit in any of the subregular classes that are commonly entertained in the literature. We show that when construed as a phonotactic dependency, the process fits into a class we call *input-output tier-based strictly local* (IO-TSL), a natural extension of the familiar class TSL. IO-TSL increases the power of TSL's tier projection function by making it an input-output strictly local transduction. Assuming that /n/-retroflexion represents the upper bound on the complexity of segmental phonology, this shows that all of segmental phonology can be captured by combining the intuitive notion of tiers with the independently motivated machinery of strictly local mappings.

1 Introduction

Subregular phonology seeks to identify proper subclasses of the finite-state languages and transductions that are sufficiently powerful for natural language phenomena (see Heinz 2018 and references therein). In addition to establishing tighter bounds on cross-linguistic variation, many of these subclasses are also efficiently learnable in the limit from positive text (Heinz et al., 2012; Jardine and McMullin, 2017).

Sanskrit /n/-retroflexion, also called *nati*, is noteworthy because it has been known for a long time to be subregular but to occupy a very high position in the subregular hierarchy when construed as a phonotactic dependency (Graf, 2010; Jardine, 2016). Its singularly high complexity stems from the combination of a locally specified target (/n/ immediately before a sonorant) with both a non-local trigger (a preceding retroflex) and three independent blocking effects, one of which is itself subject to blocking. Established classes such as *strictly local* (SL) and its extension *tier-based strictly local* (TSL; Heinz et al., 2011) cover a wide range of phonological phenomena, yet they provably cannot enforce the phonotactic conditions of *nati*.

However, as we show in this paper, *nati* can be handled by a natural extension of TSL. In TSL, a tier projection function masks out all segments that do not belong to some specified subset of the alphabet. This allows for simple non-local dependencies to be regulated in a local fashion. More involved patterns can be accommodated by increasing the complexity of the tier projection. In order to capture *nati*, the projection function has to consider two factors when choosing whether or not to project a symbol: I) the local context in the string, and II) which symbols are already on the tier. This makes it a special case of input-output strictly local maps, which is why we call this extended version of TSL *input-output TSL* (IO-TSL).

IO-TSL is a natural extension of TSL — it subsumes it as a special case and expands on recent proposals to make tier projection structure-sensitive. De Santo and Graf (2017) propose input strictly local maps to handle certain cases noted as problematic for TSL in McMullin (2016), and similar proposals are made in Baek (2017) and Yang (2018) for phonology and Vu et al. (2018) for syntax. Mayer and Major (2018), on the other hand, suggest based on Graf (p.c.) that backness harmony in Uyghur is TSL with output strictly local tier projection; Graf and Shafiei (2018) apply the same idea to syntax. Input-output strictly local projection merely combines these two extensions.

The paper is laid out as follows. We first introduce TSL (§2.1) and subsequently generalize it to IO-TSL (§2.2), some properties of which are discussed in §2.3. The empirical facts of *nati* are presented in §3 based primarily on Ryan (2017), followed by our IO-TSL analysis in §4.

Proceedings of the 15th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology, pages 151–160

https://doi.org/10.18653/v1/P17

2 Defining IO-TSL

2.1 TSL

Throughout the paper, we use ε to denote the empty string, S^* for the Kleene closure of S, and S^+ for S^* without the empty string. We use S^k to denote the proper subset of S^* that only contains strings of length k, and we write s^k as a shorthand for $\{s\}^k$.

Let Σ be some fixed alphabet and $s \in \Sigma^*$. The set $f_k(s)$ of *k-factors* of s consists of all the length-k substrings of $\rtimes^{k-1} s \ltimes^{k-1}$, where $\rtimes, \ltimes \notin \Sigma$ and $k \geq 1$.

Definition 1. A stringset $L \subseteq \Sigma^*$ is *strictly k-local* (SL-k) iff there is some $G \subseteq (\Sigma \cup \{\rtimes, \ltimes\})^k$ such that $L = \{s \in \Sigma^* \mid f_k(s) \cap G = \emptyset\}$.

Intuitively, G defines a *grammar* of forbidden substrings that no well-formed string may contain. The class SL of strictly local stringsets is $\bigcup_{k \geq 1}$ SL-k.

Example 1. The string language $(ab)^+$ is generated by the grammar $G := \{\rtimes\ltimes, \rtimes b, aa, bb, a\ltimes\}$ and thus is SL-2. For instance, aba is illicit because $f_2(aba) \cap G = \{a\ltimes\} \neq \emptyset$, whereas $f_2(abab) \cap G = \emptyset$.

For every $T \subseteq \Sigma - \{\varepsilon\}$, a *simple tier projection* π_T is a transduction that deletes all symbols not in T:

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

Definition 2. A stringset $L \subseteq \Sigma^*$ is *tier-based strictly k-local* (TSL-k) iff there exists a $T \subseteq \Sigma - \{\varepsilon\}$ and an SL-k language $K \subseteq T^*$ such that $L := \{s \in \Sigma^* \mid \pi_T(s) \in K\}$. It is TSL iff it is TSL-k for some k.

TSL languages are string languages that are SL once one masks out all irrelevant symbols.

Example 2. Consider all strings over $\{a, b, c\}$ that contain exactly one b and exactly one c. This language is TSL-3: let $T := \{b, c\}$, and $K := \{bc, cb\}$, which is an SL-3 language (the reader is invited to write down the grammar for K). The licit string $aabac$, for instance, is first projected to bc, which is a member of K. The illicit $aaba$, on the other hand, is projected to $b \notin K$.

2.2 IO-TSL

The power of TSL can be increased by changing the nature of the tier projection π. In particular, it can be generalized to strictly local maps (Chandlee, 2014). Due to space constraints, we immediately define input-output strictly local projections without discussing the earlier work on subregular mappings on which our idea builds. The interested reader should consult Chandlee (2014, 2017) and Chandlee and Heinz (2018).

An (i, j)-*context* c is a 4-tuple $\langle \sigma, b, a, t \rangle$ with $\sigma \in \Sigma$, t a string over $\Sigma \cup \{\rtimes\}$ of length $j - 1$, and a and b strings over $\Sigma \cup \{\rtimes, \ltimes\}$ of combined length $i - 1$. The context specifies that σ should be projected whenever both of the following hold: it occurs between the substrings b (look-back) and a (look-ahead), and the tier constructed so far ends in t. The value of i determines the size of the input window, which includes the look-ahead and look-back spans, as well as the symbol itself. The value of j indicates how far back along the tier we can look, including the current symbol. Given a set of contexts $c_1, c_2, \ldots, c_n$, we call it an (i, j)-*context set* $C(i, j)$ iff for every c_m $(1 \leq m \leq n)$ there are $i_m \leq i$ and $j_m \leq j$ such that c_m is an (i_m, j_m)-context.

Note that in a context set $C(i, j)$, i and j refer to the *maximum* input and output window sizes considered by any (i, j)-context. The individual (i, j)-contexts may vary in size within these bounds. This is merely a matter of notational convenience and does not affect generative capacity.

Definition 3. Let C be an (i, j)-context set. Then the *input-output strictly* (i, j)*-local* (IOSL-(i, j)) tier projection π_C maps every $s \in \Sigma^*$ to $\pi'_C(\rtimes^i \star s \ltimes^i, \rtimes^j)$, where $\pi'_C(ub \star \sigma av, wt)$ is

$$\begin{cases} \varepsilon & \text{if } \sigma av = \varepsilon, \\ \sigma \pi'_C(ub\sigma \star av, wt\sigma) & \text{if } \langle \sigma, b, a, t \rangle \in C, \\ \pi'_C(ub\sigma \star av, wt) & \text{otherwise.} \end{cases}$$

for $\sigma \in \Sigma$ and $a, b, t, u, v, w \in (\Sigma \cup \{\rtimes, \ltimes\})^*$.

The first argument to π'_C is the input string, with $\star$ as a diacritic to mark the position up to which the string has been processed. The second argument contains the symbols that have already been projected. A schematic diagram of the projection function π_C that arises from π'_C is shown in Fig. 1.

Example 3. Let $\Sigma := \{a, b, c\}$ and consider the tier projection that always projects the first and last symbol of the string, always projects a, never projects c, and projects b only if the previous symbol on the tier is a. This projection is IOSL-(2,2). The context set contains all the contexts below, and only those:

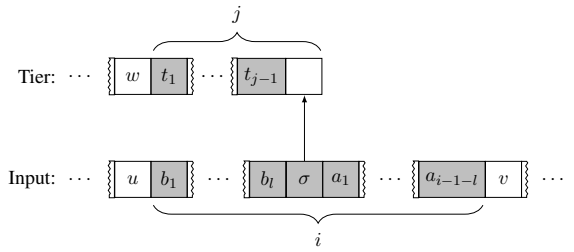

Figure 1: The projection function π_C. Grey cells indicate symbols in the input and tier strings that are considered when deciding whether to project σ.

- $\langle \sigma, \rtimes, \varepsilon, \varepsilon \rangle$ for all $\sigma \in \Sigma$,
- $\langle \sigma, \varepsilon, \ltimes, \varepsilon \rangle$ for all $\sigma \in \Sigma$,
- $\langle a, \varepsilon, \varepsilon, \varepsilon \rangle$,
- $\langle b, \varepsilon, \varepsilon, a \rangle$.

The first two of these contexts ensure that any segment is projected if it occurs at the beginning of the string or the end of the string. The third context ensures that a is always projected as all occurrences of a will be trivially preceded and followed by ε in the input and preceded on the tier by ε. The final context ensures that b is projected regardless of what precedes or follows in the input, but only if the previous symbol on the tier is a. Given the previous constraints, this is equivalent to saying that b is only projected if it is the first b encountered after seeing an a earlier in the string.

Definition 4. A stringset $L \subseteq \Sigma^*$ is *input-output tier-based strictly (i, j, k)-local* (IO-TSL-(i, j, k)) iff there exists an IOSL-(i, j) tier projection π_C and an SL-k language K such that $L := \{s \in \Sigma^* \mid \pi_C(s) \in K\}$. It is IO-TSL iff it is IO-TSL-(i, j, k) for some i, j, and k.

Note that TSL-k is identical to IO-TSL-$(1, 1, k)$, which shows that IO-TSL is indeed a generalization of TSL.

2.3 Some properties of IO-TSL

It is fairly easy to show that IO-TSL languages are definable in first-order logic with precedence and hence star-free. We conjecture that IO-TSL is in fact a proper subclass of the star-free languages.

Conjecture 1. *IO-TSL $\subsetneq$ Star-Free.*

Consider the star-free string language $L := aL'a \cup bL'b$ where L' is $(d^+cd^+ed^+)^+$. In order to ensure the long-distance alternation of c and e, one has to project every c and every e, and in order to ensure the matching of the first and last segment those have to be projected too. But then the set of well-formed tiers is $a(ce)^+a \cup b(ce)^+b$, which is not in SL because it violates suffix substitution closure (cf. Heinz, 2018). Hence L is not IO-TSL (although it is in the intersection closure of TSL). A fully worked out proof would have to show that all other IOSL tier projections fail as well.

Like most subregular language classes, IO-TSL is not closed under relabeling. This follows from the familiar insight that $(aa)^+$, which isn't even star-free, is a relabeling of the SL-2 language $(ab)^+$. We state a few additional conjectures without further elaboration.

Conjecture 2. *IO-TSL is not closed under intersection, union, relative complement, or concatenation.*

Conjecture 3. *IO-TSL is incomparable to the following classes:*

- *locally threshold testable languages (LTT),*
- *locally testable (LT),*
- *piecewise testable (PT),*
- *interval-based strictly piecewise (IBSP; Graf, 2017, 2018)*
- *strictly piecewise (SP; Rogers et al., 2010)*

If correct, these properties of IO-TSL would mirror exactly those of TSL, further corroborating our claim that IO-TSL is a very natural generalization of TSL.

That said, IO-TSL is a fair amount more complex than TSL. In the next section, we discuss the empirical facts of Sanskrit /n/-retroflexion that motivate the introduction of this additional complexity.

3 Sanskrit n-retroflexion

Sanskrit /n/-retroflexion, also called *nati*, has been studied extensively throughout the history of linguistics, and has received particularly close scrutiny within generative grammar. The notorious complexity of the phenomenon is the product of the interaction of multiple (individually simple) conditions: long-distance assimilation (§3.1), blocking by preceding coronals (§3.2), mandatory adjacency to sonorants (§3.3), blocking by preceding plosives (§3.4), and blocking by following retroflexes (§3.5).

Even a cursory look at the previous literature is beyond the scope of this paper, so we refer the reader to Ryan (2017) for a detailed literature review and analysis of the phenomenon. We draw data from Müller (1886), Hansson (2001), and Ryan (2017) and use the transcription conventions from Ryan (2017). Page numbers for the sources are indicated in the table captions of the data.

3.1 Base pattern

The central aspect of *nati* is simple: underlyingly anterior /n/ becomes retroflex [ɳ] when it is preceded in the word by a non-lateral retroflex continuant (one of /ɻ/, /ɻ̈/, /ɻ̣/, or /ʂ/). The retroflex trigger can occur arbitrarily far to the left of the nasal target. Tables 1 and 2 respectively show the alternations in the instrumental singular suffix /-eːna/ when attached to roots without and with a trigger. Triggers, blockers, and targets are bolded in all tables.

Form	**Gloss**
káːm-eː**n**a	‘by desire’
baːɳ-eː**n**a	‘by arrow’
muːɖ$^{\text{ɦ}}$-eː**n**a	‘by the stupid (one)’
joːg-eː**n**a	‘by means’

Table 1: Forms with no *nati* (Ryan, 2017, p. 305)

Form	**Gloss**
na**ɻ**-eː**ɳ**a	‘by man’
manu**ʂ**j-eː**ɳ**a	‘by human’
ɻaːg$^{\text{ɦ}}$avɔ-eː**ɳ**a	‘by the Rāghava’
pu**ʂ**paug$^{\text{ɦ}}$-eː**ɳ**a	‘by the heap of flowers’
b**ɻ**afima**ɳ**ja	‘kind to Brahmans’
ni**ʂ**a**ɳɳ**a	‘rested’
ak**ʂ**a**ɳ**ʊat	‘having eyes’

Table 2: Basic *nati* examples (Ryan, 2017, p. 305 and Müller, 1886, p. 44)

Viewing *nati* as a phonotactic phenomenon rather than a mapping from underlying representations to surface forms, we can formalize it as the constraint that no [n] may appear in the context $R \cdots \_$, where R is a non-lateral retroflex continuant. This does not constitute an analysis, but it clarifies the formal character of the process. As we will see in the remainder of this section, though, the context is in fact much more complicated than just $R \cdots \_$.

3.2 Unconditional blocking by intervening coronals

If a coronal segment (including retroflexes) occurs between the trigger and the target, then *nati* is blocked. The only exception to this is the palatal glide /j/ (cf. Table 2) — this is an important point that we will return to in §4.3. Crucially, [ɳ] itself is a coronal blocker, meaning the assimilation process only affects the first in a series of eligible targets. An exception to this is geminate /nn/ sequences, where both instances of /n/ undergo *nati* (cf. Table 2; this complication will also be discussed in §4.3). Examples of *nati* blocked by an intervening coronal are shown in Table 3. Leaving aside geminates for the moment, the illicit context for [n] now becomes $R\overline{C}^*\_$, where $\overline{C}$ matches [j] and all segments that are not coronals.

Form	**Gloss**
ɻáth-eː**n**a	‘by chariot’
gaɻu**ɖ**-eː**n**a	‘by Garuḍa’
pɻa-**ɳ**i**n**aːja	‘lead forth’
vaɻ**ɳ**-a**n**aːnam	*no gloss*

Table 3: Intervening coronal blocking (Hansson, 2001, p. 227 and Ryan, 2017, p. 305)

3.3 Mandatory adjacency to sonorant

Next, the /n/ must be immediately followed by a non-liquid sonorant to undergo *nati*. More precisely, the following symbol must be a vowel, a glide, /m/, or /n/ itself (Whitney, 1889). No other nasals can occur following /n/ due to independent phonotactic constraints in the language (Emeneau, 1946). Like the special status of /j/ and geminates, this will become important in §4.3 but can be ignored for now.

Examples of cases where *nati* is blocked by the following sound, or lack thereof, are shown in Table 4. Again updating the illicit context for [n], we get $R\overline{C}^*\_S$, where S is a vowel, glide, /m/, or /n/.

Form	**Gloss**
bɻafima**n**	‘brahman’
t**ɻ=n=t**-te	‘split (3Pl middle)’
ca**ɻ**-a-**n**-ti	‘wander (3Pl)’

Table 4: Blocking when no non-liquid sonorant follows (Hansson, 2001, p. 229 and Ryan, 2017, p. 318)

3.4 Conditional blocking by preceding velar and labial plosives

In addition to coronals blocking when they intervene between the trigger and the target, velar and labial plosives can also block *nati*, but only when two conditions are met at the same time: I) the plosive occurs immediately before the target, and II) a left root boundary $\surd$ intervenes between the target and trigger. Left root boundaries are generally omitted for clarity when they occur at the left edge of a word. Table 5 shows that left root boundaries alone are not sufficient to block *nati*. Table 6 shows cases where a labial or velar plosive blocks *nati* across a left root boundary, and Table 7 shows cases where such a plosive does not block because no left root boundary intervenes.

Form	**Gloss**
pɻa-$\surd$**ɳ**aɕ-ja-ti	'vanishes (3s)'
ʊɻtɻa-$\surd$fiá**ɳ**a	'Vr̥tra-killing'
pɻa-$\surd$mi:**ɳ**-a:-ti	'frustrates (3s)'

Table 5: *Nati* occurring across left root boundaries (Ryan, 2017, pp. 320, 321, 324)

Form	**Gloss**
pɻ-$\surd$a:**p**-**n**o:-ti	'attains (3s)'
pɻ-$\surd$a:**p**-**n**u-a:h	'should attain (2s opt.)'
pɻa-$\surd$b$^{\text{fi}}$a**g**-**n**a	'broken'

Table 6: Labial/velar blocking with intervening root boundary (Ryan, 2017, pp. 318, 321)

Form	**Gloss**
$\surd$**ɻug-ɳá**	'break (pass. part.)'
$\surd$t**ɻ̣p-ɳ**V-	'be satisfied (pres. stem)'
$\surd$**ɻé:kɳ**as	'inheritance'

Table 7: No labial/velar blocking (Ryan, 2017, p. 319)

To accommodate these new facts, we change the context to $R\alpha\_S$. Here α is any string that does not contain a coronal and does not match $\cdots\surd\cdots P$, where P is a velar plosive or a labial plosive.

3.5 Conditional blocking by following retroflex

There is one final complication: *nati* is blocked when a retroflex occurs somewhere to the right of the target /n/. Like with blocking by plosives, though, two conditions must be met: I) as above, a left root boundary separates the trigger and the target of *nati*, and II) no coronal intervenes between /n/ and the blocking retroflex (so coronals "block" retroflex blocking). Keep in mind that, as in §3.2, /n/ itself is coronal and thus can act as a blocker, a point that will be important in §4.

Table 8 shows cases where a following retroflex blocks in the presence of a left root boundary intervening between trigger and target, and Table 9 shows cases where it does not block because a coronal intervenes between the target and the blocker, or no intervening boundary is present. Note in the final example of Table 9 that an intervening left root boundary between the trigger and the blocker has no effect — the boundary must occur before the targeted /n/.

Ryan (2017) notes that it is unclear from the data whether the retroflex blocking is truly unbounded, or if the blocker must occur within a certain distance of the target. We assume the pattern is unbounded here, but it does not significantly alter the analysis if this is not the case.

Form	**Gloss**
pɻa-$\surd$**n**a**ʂ**-**ʈ**um	'to vanish (inf.)'
pɻa-$\surd$**nɻ̣**t-	'dance forth'
pɻa-$\surd$**n**ak**ʂ**-	'approach'

Table 8: Following retroflex blocking (Ryan, 2017, p. 325)

Form	**Gloss**
pɻa-$\surd$**ɳ**eː-**tɻ̣**	'leader'
$\surd$bɻa:fima**ɳ**-éː-**ʂ**u	'Brahmins (loc. pl.)'
$\surd$pɻ-**ɳ**a-k-**ʂ**i	'unite (2s)'
$\surd$puɻa:**ɳ**a-$\surd$**ʂ**i	'ancient rishi'

Table 9: No following retroflex blocking (Ryan, 2017, p. 325, 326)

With this last piece of data in place, we can finally give a full description of *nati* as a phonotactic constraint on the distribution of [n] in surface forms.

Definition 5 (nati)**.** No [n] may occur in a context $R\alpha\_S\beta$ such that the following hold:

- R is a non-lateral retroflex continuant, and
- S is a vowel, glide, /m/, or /n/ and
- none of the following blocking conditions are met:

- α contains a coronal C, or
- α matches $\cdots \surd \cdots P$, where P is a velar plosive or labial plosive, or
- α contains $\surd$, and β contains a retroflex that is not preceded by a coronal in β.

This description can be translated into a first-order formula with precedence to show that *nati* is star-free. In the next section, we show that it is also IO-TSL.

4 Formal analysis

The IO-TSL analysis of *nati* is a bit convoluted, but more straightforward than one might expect. All the heavy lifting is done by the tier projection mechanism (§4.1). In any case where the projection considers the context at all, it uses a look-ahead of 1 or 2, a look-back of 1 in the string, a look-back of 1 on the tier, or a mixture of these options. In particular, P and C have complex tier projection conditions. Our projection function creates tiers of a very limited shape that are easily shown to be SL-3 (§4.2). While our analysis uses abstract symbols such as R, P, and C, the complexity of *nati* remains the same even if one talks directly about the relevant segments instead (§4.3).

4.1 Tier projection

As contexts make for a verbose description, we opt for a more informal specification of the IOSL tier projection. The projection rules for each symbol are sufficiently simple that this does not introduce any inaccuracies.

An IOSL-$(3, 2)$ tier projection for *nati* is shown below. For each condition we list its individual complexity. Note that the rules below merely specify how tiers are constructed, not which tiers are well-formed. This is left for §4.2.

- Always project R. *IOSL-(1,1)*
- Project S if it is immediately preceded by [n] in the input. *IOSL-(2,1)*
- Project $\surd$ if the previous tier symbol is R. *IOSL-(1,2)*
- Project P if the previous tier symbol is $\surd$ and the next two input symbols are [n] and S. *IOSL-(3,2)*
- Project C if the previous tier symbol is R, $\surd$, or S, unless C is [n] and the next input symbol is S. *IOSL-(2,2)*
- Project every retroflex (not just those matching R) if the previous tier symbol is S. *IOSL-(1,2)*
- Don't project anything else. *IOSL-(1,1)*

Table 10 shows variations of the previous data points with the tiers projected according to the rules above.

Let us briefly comment on the intuition behind these projection rules. We always want to project R since this is the only potential trigger for *nati*. For [n], we do not want to project all instances as this may end up restricting the distribution of an [n] that is not a suitable target anyways. In addition, we do not want to project [n] itself as this would make it impossible to distinguish an [n] that was projected as a potential *nati*-target from one that was projected as an instance of C. So instead, we project the sonorant after [n]. As a sonorant has no other reason to appear on the tier, it can act as an indirect representative of an [n] that may be targeted by *nati*.

The left root boundary $\surd$ matters only if it occurs between R and [n]. Since we build tiers from left-to-right, we cannot anticipate the presence of [n] on the tier, and in the input string there is no upper bound on how far [n] might be from $\surd$. Hence we have to project $\surd$ after every R, even if R does not end up triggering *nati*. A redundant instance of $\surd$ on the tier is no problem.

As for P, its presence matters only when it occurs before a potential *nati* target, so we project it only in these configurations. We also impose the requirement that the previous tier symbol is $\surd$ as P needs a left root boundary between R and [n] to become a blocker. This kind of mixing of input and output conditions is not necessary for P, but it is essential for C.

The coronal C is the strongest blocker. In contrast to $\surd$, it does not depend on other material in the string, so it should be projected not only immediately after R but also if the previous tier symbol is $\surd$ (from which we can infer that the tier symbol before that is R). A C between [n] and a retroflex inhibits the latter's ability to block *nati*, so we also need to project C if the previous tier symbol is S (our tier stand-in for [n]). As arbitrary retroflex segments are projected only if the previous tier symbol is S, projecting C after S effectively blocks projection of retroflexes. Note that we do not project C when it is an [n] before

S as this is a potential *nati*-target and hence S is projected anyways.

4.2 SL grammar

Once the IOSL tier projection is in place, specifying the forbidden substrings is a simple task. Consider a segment [n] that is followed by S and hence a potential target for *nati*. If there are any R that can trigger *nati*, it suffices to consider the closest one. Given such a configuration $R \cdots$ [n]S, any tier will have the shape below:

$$\cdots R\ (C \mid \surd(C \mid P))\ S\ (C \mid S \mid \text{retroflex}) \cdots$$

Here $(X \mid Y)$ means "X or Y or nothing".

Based on this abstract template, the following substrings are illicit because they indicate an [n] in a configuration where *nati* should apply:

- $R\ S$, and
- $R\ \surd\ S\ X$ (where X is $\ltimes$, C, or S).

That these are the only four substrings that need to be forbidden is illustrated in Table 10. But note that $\surd$ is always immediately preceded by R on the tier, so the illicit substrings for the second case can be shortened to $\surd\ S\ \ltimes$, $\surd\ S\ C$, and $\surd\ S\ S$. Therefore the longest forbidden substring has at most 3 segments and the dependencies over the tier are SL-3. As a result, *nati* is IO-TSL-(3,2,3); these surprisingly low thresholds suggest that *nati* is still fairly simple from a formal perspective.

4.3 Removing abstract symbols

There are still a few minor points that merit addressing. As noted in §2.2, IO-TSL is not closed under relabeling, so the fact that the abstract patterns used in the previous sections are IO-TSL does not imply that *nati* itself is. This is the case only if one can compile out the abstract symbols R, S, C, P, and retroflex into specific segments like [n], [j], and [ɻ]. No problems arise in cases where no segment corresponds to more than one abstract symbol. For example, [j] only matches S. If [j] were also a coronal blocker, then it would also match C and the account in the previous section would no longer work since it would not be clear if a [j] on a tier represents a blocker C or a sonorant S. In such a case, the use of abstract symbols would simplify the pattern in an illicit way. There are two instances of overlap in our analysis: R versus arbitrary retroflexes, and [n] belonging to both C and S.

Regarding the split between R and arbitrary retroflexes, it is actually unclear from the data whether the two are distinct classes. The available data includes only instances of segments matching R acting as blockers, though Ryan (2017) suggests based on his analysis that all retroflexes should be able to block. But even if the two are distinct, that is unproblematic for our account. The projection of R is less restricted than that of arbitrary retroflexes as the latter are only projected if the previous tier symbol is S. This discrepancy matters only in cases where a C-segment occurs after [n]. In this case, arbitrary reflexives do not project whereas R still does. Projecting an R-segment after a C has no consequences, though. If the preceding substring matches $\surd\ SC$, then projecting R won't salvage it. If it does not match this pattern, projecting R won't make the tier ill-formed. Hence the minimal difference between R and arbitrary retroflexes — if it exists — is immaterial for our account.

That [n] belongs to both C and S is not much of an issue, either. Since S is projected only after [n], an [n] on the tier could be an instance of S only if there are [nn] clusters targeted by *nati*. In such cases, the entire cluster undergoes *nati*. For example, we see [niʂaɳɳa], not *[niʂaɳna] (cf. Table 2). There are two solutions to this. One option is to treat these not as [nn] clusters, but rather as a single segment [nː]. As long as the projection of S is generalized to be sensitive to both [n] and [nː], this data is handled correctly by our analysis. Alternatively, we could rely on the fact that [nn] clusters in our data are always followed by S. Hence we can limit S to vowels, glides, and [m] and still end up with a sonorant on the tier after each potential *nati*-target. Either way there are again safe ways to deal with the apparent overlap between the abstract symbols.

We find it interesting in connection to this that [j] is both sonorant and coronal but does not act as a coronal blocker. If it did, it would belong to both C and S, without an easy fix to rescue our analysis. Although Ryan (2017) accounts for the special status of [j] on the basis of its articulatory properties, the fact that its behavior is also predictable from a computational perspective is intriguing. Be that as it may, the important point is that the account in §4.1 and §4.2 captures *nati* even if the abstract symbols are compiled out to the actual segments.

Example	Tier	Forbidden substring	Example	Tier	Forbidden substring
naɻeːŋa	aɻŋ⋉	–	√ɻugŋá	ɻŋ⋉	–
*naɻeːna	aɻa⋉	ɻa	*√ɻugná	ɻá⋉	ɻá
pɻaŋinaːja	ɻŋa⋉	–	pɻa√ŋeːt̪	ɻ√n̪t̪⋉	–
caɻanti	ɻt⋉	–	*pɻa√neːt̪	ɻ√eːt̪t̪⋉	√eːt
ʊɻtɻa√ɦáŋa	ɻtɻ√ŋ⋉	–	√bɻaːɦmaŋéːʂu	ɻŋʂ⋉	–
*ʊɻtɻa√ɦána	ɻtɻ√a⋉	√a⋉	*√bɻaːɦmanéːʂu	ɻéːʂ⋉	ɻéː
pɻ√aːpnoːti	ɻ√poːt⋉	–	pɻa√nakʂ	ɻ√aʂ⋉	–

Table 10: Selected data points and ungrammatical variants from Tables 2 to 9 with their tiers and forbidden subsequences (if any); ⋉ is added to each tier for the sake of exposition.

4.4 Relevance and interpretation

Our analysis establishes IO-TSL as a tighter upper bound on the complexity of *nati* when construed as a single phonotactic constraint over strings. This does not mean, though, that this is the only viable view of *nati*, as it could have been analyzed as a collection of individually simple constraints (cf. Ryan, 2017), a condition on graph structures in the sense of Jardine (2017), or a mapping from underlying representations to surface forms. These are all insightful perspectives, but they are orthogonal to our goal of bounding the overall complexity of *nati*. Our finding that *nati* (and probably segmental phonology as a whole) is IO-TSL is analogous to claims that syntax yields mildly context-sensitive string languages (Joshi, 1985) even though the actual representations are trees with a tree-to-string mapping. The IO-TSL nature of *nati* provides a rough complexity baseline on which more nuanced and linguistically insightful notions of complexity can be built.

We do find it interesting that IO-TSL as a natural generalization of TSL is sufficient to capture *nati*. But IO-TSL is still too liberal an upper bound. Just like the TSL-extensions used in Baek (2017), De Santo and Graf (2017), and Yang (2018), IO-TSL allows for unattested phenomena such as first-last harmony (Lai, 2015). Future work may identify subclasses of IO-TSL that allow for *nati* but not first-last harmony. IO-TSL in its current form is nonetheless a fairly restrictive upper bound on *nati*.

One final point of contention is our treatment of *nati* as a long distance process, rather than local retroflex spreading as suggested by Ryan (2017). Unfortunately, there is no clear evidence that the posited local spreading is visible in the output string. Spreading of unpronounced material would be an instance of *feature coding*, which destroys subregular complexity because every regular language can be made SL-2 this way (see e.g. Rogers 1997). Ryan's analysis may still be more appropriate from a linguistic perspective, but for our purposes it may incorrectly nullify the complexity of *nati*.

5 Conclusion

We have shown that even a highly complex process like *nati* can be regarded as a local dependency over tiers given a slightly more sophisticated tier projection mechanism that considers both the local context in the input and the preceding symbol(s) on the tier. This extension is natural in the sense that it co-opts mechanisms that have been independently proposed in the subregular literature as a more restricted model of rewrite rules. Moreover, the complexity is fairly low as *nati* fits into IO-TSL-$(3, 2, 3)$. A careful reanalysis of the data may be able to lower these thresholds even more by incorporating independent restrictions on the distribution of some segments. Allowing the tier projection to proceed from right to left might also affect complexity.

The effect of the increased power on learnability is still unknown. IO-TSL-(i, j, k) is a finite class given upper bounds on i, j, and k, which immediately entails that the class is learnable in the limit from positive text (Gold, 1967). This leaves open whether the class is efficiently learnable, as is the case for TSL (Jardine and McMullin, 2017) and the strictly local maps IO-TSL builds on (Chandlee et al., 2014, 2015). But IO-TSL adds two serious complications: the learner does not have access to the output of the tier projection function in the training data, and inferring correct contexts presupposes correctly built tiers.

Acknowledgments

We would like to thank David Goldstein and Canaan Breiss for helping us better understand *nati*.

References

Hyunah Baek. 2017. Computational representation of unbounded stress: Tiers with structural features. Ms., Stony Brook University; to appear in *Proceedings of CLS 53*.

Jane Chandlee. 2014. *Strictly Local Phonological Processes*. Ph.D. thesis, University of Delaware.

Jane Chandlee. 2017. Computational locality in morphological maps. *Morphology*, pages 599–641.

Jane Chandlee, Rémi Eyraud, and Jeffrey Heinz. 2014. Learning strictly local subsequential functions. *Transactions of the Association for Computational Linguistics*, 2:491–503.

Jane Chandlee, Rémi Eyraud, and Jeffrey Heinz. 2015. Output strictly local functions. *Mathematics of Language*.

Jane Chandlee and Jeffrey Heinz. 2018. Strict locality and phonological maps. *Linguistic Inquiry*, 49:23–60.

Aniello De Santo and Thomas Graf. 2017. Structure sensitive tier projection: Applications and formal properties. Ms., Stony Brook University.

M.B. Emeneau. 1946. The nasal phonemes of Sanskrit. *Language*, 22(2):86–93.

E. Mark Gold. 1967. Language identification in the limit. *Information and Control*, 10:447–474.

Thomas Graf. 2010. Comparing incomparable frameworks: A model theoretic approach to phonology. *University of Pennsylvania Working Papers in Linguistics*, 16(2):Article 10.

Thomas Graf. 2017. The power of locality domains in phonology. *Phonology*, 34:385–405.

Thomas Graf. 2018. Locality domains and phonological c-command over strings. To appear in *Proceedings of NELS 2017*.

Thomas Graf and Nazila Shafiei. 2018. C-command dependencies as TSL string constraints. Ms., Stony Brook University.

Gunnar Ólafur Hansson. 2001. *Theoretical and Typological Issues in Consonant Harmony*. Ph.D. thesis, University of California, Berkeley.

Jeffrey Heinz. 2018. The computational nature of phonological generalizations. In Larry Hyman and Frank Plank, editors, *Phonological Typology*, Phonetics and Phonology, chapter 5, pages 126–195. Mouton De Gruyter.

Jeffrey Heinz, Anna Kasprzik, and Timo Kötzing. 2012. Learning in the limit with lattice-structured hypothesis spaces. *Theoretical Computer Science*, 457:111–127.

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.

Adam Jardine. 2016. Computationally, tone is different. *Phonology*, 33:247–283.

Adam Jardine. 2017. The local nature of tone-association patterns. *Phonology*, 34:385–405.

Adam Jardine and Kevin McMullin. 2017. Efficient learning of tier-based strictly k-local languages. In *Proceedings of Language and Automata Theory and Applications*, Lecture Notes in Computer Science, pages 64–76, Berlin. Springer.

Aravind Joshi. 1985. Tree-adjoining grammars: How much context sensitivity is required to provide reasonable structural descriptions? In David Dowty, Lauri Karttunen, and Arnold Zwicky, editors, *Natural Language Parsing*, pages 206–250. Cambridge University Press, Cambridge.

Regine Lai. 2015. Learnable vs unlearnable harmony patterns. *Linguistic Inquiry*, 46:425–451.

Connor Mayer and Travis Major. 2018. A challenge for tier-based strict locality from Uyghur backness harmony. In *Formal Grammar 2018. Lecture Notes in Computer Science, vol. 10950*, pages 62–83. Springer, Berlin, Heidelberg.

Kevin McMullin. 2016. *Tier-Based Locality in Long-Distance Phonotactics: Learnability and Typology*. Ph.D. thesis, University of British Columbia.

Friedrich Max Müller. 1886. *A Sanskrit Grammar for Beginners: In Devanagari and Roman Letters*. Longmans, Green, and Co., London.

James Rogers. 1997. Strict LT_2 : Regular :: Local : Recognizable. In *Logical Aspects of Computational Linguistics: First International Conference, LACL '96 (Selected Papers)*, volume 1328 of *Lectures Notes in Computer Science/Lectures Notes in Artificial Intelligence*, pages 366–385. Springer.

James Rogers, Jeffrey Heinz, Gil Bailey, Matt Edlefsen, Molly Vischer, David Wellcome, and Sean Wibel. 2010. On languages piecewise testable in the strict sense. In Christan Ebert, Gerhard Jäger, and Jens Michaelis, editors, *The Mathematics of Language*, volume 6149 of *Lecture Notes in Artificial Intelligence*, pages 255–265. Springer, Heidelberg.

Kevin Ryan. 2017. Attenuated spreading in Sanskrit retroflex harmony. *Linguistic Inquiry*, 48(2):299–340.

Mai Ha Vu, Nazila Shafiei, and Thomas Graf. 2018. Case assignment in TSL syntax: A case study. Ms., Stony Brook University and University of Delaware.

William Dwight Whitney. 1889. *Sanskrit Grammar*. Oxford University Press, London.

Su Ji Yang. 2018. Subregular complexity in Korean phonotactics. Undergraduate honors thesis, Stony Brook University.