The computational nature of templatic morphology

Hossep Dolatian, Jon Rawski, Sedigheh Moradi (Stony Brook University)

Keywords: semitic, template, computational complexity, finite state morphology, subregular

Introduction & Contribution: Non-concatenative morphology in Semitic languages has been modeled with different types of computational machinery (Kiraz, 2001) with unclear differences in power. We analyze the computational properties of templatic morphology and determine the minimal computational power needed. We show that templatic morphology requires four different levels of power depending on various factors:

Example	Needed power	Is the template known in advance?	Is there spreading?	Is there pre-association?
katab	ISL over graph	✓	✗	✗
katab, ktub	ISL over multiple tape	✗	✗	✗
kattab, ktabab	OSL over multiple tape	✗	✓	✗
ta-kattab	OSL over a graph	✗	✓	✓

Background: In Semitic languages, a word can consist of three morphological items (McCarthy, 1981), e.g., an abstract root *R=ktb*, a vocalism *V=a*, and a template *T=CVCVC*. The three items are combined to form one word *katab*. Others can be formed with different templates (Table 1). The number of templates in a language like Arabic is finite and relatively small. Some templates like *CVCVC* in *katab* have no segments pre-associated to any slot, while some templates like *tV-CVCCVC* in *ta-kattab* have specific segments *s,t* associated to the first two C slots. Some like *kattab* or *ktabab* involve spreading or the multiple association of a consonant or vowel to multiple slots in the template.

Finite-state morphology: Morphological processes can be modeled using finite-state automata and transducers. Thus, the regular class of languages and functions provides a sufficient computational upper-bound. However, many processes don't require the full power of regular transductions, and can be captured by subclasses of finite-state machines, i.e. they are *subregular* (Chandlee, 2017).

katab	'he wrote'
kutib	'it was written'
ktub	'write!'
kattab	'he dictated
ta-kattab	'be dicated'

Table 1: Words derived from root *ktb*.

Most *concatenative morphology* requires only the simplest subclass of transducers: Input-Strictly-Local (ISL) functions. A function is ISL-k if at position *i* in the input, it only needs to keep track of the last k-1 input symbols in order to output some symbol *o*. For example, adding a suffix *-ed* is ISL-1 because it only needs to know where the word-final boundary is. Other processes are Output-Strictly-Local (OSL) if they require information in the output. For example, iterative nasalization as in Malay /pəŋawasan/→[pəŋãw̃ãsan] is OSL because a vowel is nasalized after a nasalized segment in the output. Functions can likewise operate over a linear string of segments on one input tape, or over multiple strings on multiple tapes as in autosegmental phonology. For non-concatenative morphology, finite-state functions are again *sufficient*, but little is known about what proper subclass of regular languages/functions is *necessary*. Using subregular morphophonology, we determine the minimal computational requirements of root-and-pattern morphology and template filling.

I. Computing a simple template: As for non-concatenative morphology, we show that for each *specific* template, a unique finite-state transducer can be made that requires only local information in the input, i.e. it is ISL. The intuition is simple: if the input template is known as *CVCVC*, and the input is {*ktb*} & {*a*} then an FST can be designed which remembers which consonant was first, second, or third. Because the number of consonants is fixed at some number *3* or *4*, and because the size of the template is fixed (i.e., we know where the first, second, or third consonants will go), then the function is ISL for k=3 or k=4. This computational result sheds light on why templatic morphology is stable across Semitic languages: the low number of consonants per root makes it computationally simple.

II. Computing any simple template: As said, for any given template, a unique ISL function can be made. The input to this function would be the set of consonants and vowels, e.g. {*ktb*} and {*a*}. However, if the template is not known in advance, then the process is no longer ISL. Specifically, if the function takes as input a set of consonants {*ktb*}, a set of vowels {*a*}, **and** a template like *CVCVC* or *CCVC*, then the FST needs to check the template and see where the second C should go: in the 3^{rd} slot for a *CVCVC* template but the 2^{nd} slot for a *CCVC* template. This information is not local in the input if the input is given as a linear string of segments and slots. However, it *is* ISL if the input is on *multiple* tapes. The intuition is that having multiple tapes gives us multiple windows of locality to work with.

III. Computing a template with spreading: Spreading occurs when there are more C slots than root consonants, or more V slots than input vowels, e.g. *kattab* and *ktabab*. Similar to Malay nasalization, this spreading process is iterative and needs access to the output. The intuition is that to produce *ktabab*, the function first outputs *ktabVC*. It then spreads the last C and V which it had outputted in order to produce *ktabab*. This need for output information makes the function be *OSL* over multiple tapes

IV. Computing a template with pre-associations: In the previous cases, there was no segment that was pre-specified in any single position, i.e. pre-associated. However, once we pre-associate segments to slots, e.g. the template *tV-CVCCVC*, the function is now OSL over a *graph*. The intuition behind this is that without pre-associations, the template tape and the consonant tape are independent of each other. However, with pre-associated edges between tapes in the input, this independence is lost. In order to compute the output *ta-kattab* with template *T=CV-CVCCVC*, root consonants {*ktb*}, and pre-associated affix consonant {*t*}, the FST now needs to operate over a richer representation, a graph.

Conclusion: Root-and-pattern morphology is a regular process but there is little work in determining the minimum computational power that it needs. This paper fills that gap and develops a hierarchy of computational power for root-and-pattern morphology using different types of finite-state transducers. This result opens doors to understanding how root-and-pattern morphology can be learned and cognitively processed, and it provides a computational typology of Semitic morphology which we will expand in future work.

References: **[1]** Chandlee, J. (2017). Computational locality in morphological maps. *Morphology*, 1–43. **[2]** Kiraz, G. A. (2001). *Computational nonlinear morphology: with emphasis on Semitic languages*. Cambridge University Press. **[3]** McCarthy, J. J. (1981). A prosodic theory of noncon-

The complexity of optimizing over strictly local constraints

Nate Koser and Adam Jardine, Rutgers University

1. Introduction For grammars in Optimality Theory (Prince and Smolensky 1993), what is the relationship between a Constraint Definition Language (de Lacy 2011) and the patterns generated by optimizing over those constraints? We show that a set of stress constraints defined as *strictly local* (SL; McNaughton and Papert 1971) and evaluated in parallel OT can generate a pattern that is *regular*; that is, properly outside the *star-free* class (SF; McNaughton and Papert 1971), which has been argued to be an upper bound for stress (Rogers et al. 2013). The pattern is an unattested "sour grapes"-like stress pattern, in which iterative foot placement occurs only if it generates a full parse. It is fully regular, demonstrating that SL patterns are not closed under optimization. Furthermore, while sour grapes has received attention in harmony and tone phenomena (Padgett 1995; Wilson 2003, 2006; McCarthy 2010; Jardine 2016), sour grapes-like stress has not previously been discussed. We take the SL class as the definition language for constraints to formalize the notion that markedness constraints are negative (Jardine and Heinz 2016).

2. Background Markedness constraints are overwhelmingly negative, meaning that they ban certain illicit structures (McCarthy and Prince 1993; Jardine and Heinz 2016). We can formalize the generalization that markedness constraints should be negative with a *Constraint Definition Language* (CDL) (Eisner 1997; Potts and Pullum 2002; de Lacy 2011) for markedness constraints. We take the SL class to be the CDL.

The SL class, which lies at the bottom of the *sub-regular hierarchy* (McNaughton and Papert 1971; Rogers and Pullum 2011) of formal languages is describable by *conjunctions of negative literals* (Rogers et al. 2013, CNLs;), or a series of statements forbidding substrings. While an extremely restrictive class, the SL languages are highly relevant to phonology, as many phonotactic patterns are SL (Heinz 2010). For example, the stress constraint *CLASH is SL, as it prohibits sequences of adjacent stressed syllables.

More formally, CNLs are of the form $\neg\ a \wedge \neg\ b \wedge \neg\ c\ ...$ where each conjunct is a contiguous piece of phonological structure that is banned by the constraint. Defined as a CNL, *CLASH is written $\neg\ \acute{\sigma}\acute{\sigma}$. So the sequence $\sigma\sigma\acute{\sigma}$ satisfies $\neg\acute{\sigma}\acute{\sigma}$ because it does not include $\acute{\sigma}\acute{\sigma}$, but $\sigma\acute{\sigma}\acute{\sigma}$ violates it. Here we restrict our stress constraints to SL complexity by selecting CNLs as the constraint definition language for markedness constraints.

However, we show that even when a CDL reduces CON to SL constraints, OT can produce unattested patterns that are properly regular. This means that, even if we have a specific CDL based on computational complexity, optimization can generate patterns beyond this level of complexity. While similar results have been formalized for OT as a model of phonology in general (Frank and Satta 1998), here we present a formal analysis of surface well-formedness patterns in OT.

3. Analysis This paper considers a CON with the following constraints: IAMB, which is violated by trochees and unary feet; $\neg\ (\acute{\sigma}\sigma) \wedge \neg\ (\acute{\sigma})$. TROCH, which is violated by iambs and unary feet; $\neg\ (\sigma\acute{\sigma}) \wedge \neg\ (\acute{\sigma})$. PARSE, which is violated by unparsed syllables; $\neg\ \breve{\sigma}$ ($\breve{\sigma}$ is used here to indicate unparsed syllables). $^*\breve{\sigma}F$, which is violated when a footed syllable follows an unfooted one; $\neg\breve{\sigma}(\sigma \wedge \neg\breve{\sigma}(\acute{\sigma}$. And $^*F\breve{\sigma}$, which is violated when an unfooted syllable follows a footed one; $\neg\sigma)\breve{\sigma} \wedge \neg\acute{\sigma})\breve{\sigma}$.

The constraints IAMB, TROCH and PARSE are familiar constraints from the stress literature. *σF and *$F\sigma$ are introduced to achieve alignment of feet. These constraints are similar to the constraints **Ft/_*σ and **Ft/*σ*_* discussed in McCarthy (2003), but are defined as CNLs here. *σF and *$F\sigma$ can place feet by militating against unparsed-syllable gaps. For example, the $^*\breve{\sigma}F$ constraint prefers a candidate $(\sigma\sigma)(\sigma\sigma)(\sigma\sigma)\sigma$ to candidates $\sigma(\sigma\sigma)(\sigma\sigma)(\sigma\sigma)$ or $(\sigma\sigma)\sigma(\sigma\sigma)(\sigma\sigma)$ for not having any unparsed syllables to the left of a foot edge. $^*\breve{\sigma}F$ prefers flush left-alignment, $^*F\breve{\sigma}$ prefers flush right-alignment.

OTWorkplace (Prince et al. 2007 2017) reveals a typology of nine languages for this set of constraints: two sour grapes languages, one stressless language, two ambiguous languages (more than one optimal output for some word lengths), and four languages that differ minimally from attested languages. The pattern under discussion here is one of the sour grapes-like languages, given here:

	1σ	2σ	3σ	4σ	5σ	6σ	7σ	
(1)	$\breve{\sigma}$	$(\acute{\sigma}\sigma)$	$\breve{\sigma}\breve{\sigma}\breve{\sigma}$	$(\acute{\sigma}\sigma)(\acute{\sigma}\sigma)$	$\breve{\sigma}\breve{\sigma}\breve{\sigma}\breve{\sigma}\breve{\sigma}$	$(\acute{\sigma}\sigma)(\acute{\sigma}\sigma)(\acute{\sigma}\sigma)$	$\breve{\sigma}\breve{\sigma}\breve{\sigma}\breve{\sigma}\breve{\sigma}\breve{\sigma}\breve{\sigma}$	...

The language builds binary-only feet to the end of the word. If this cannot be done, as in odd-numbered-syllable forms, then instead no feet are created at all. This is the sour grapes-like property of this language. Instead of having some more minimal parse of syllables to feet in odd-syllable forms, there is no parsing of syllables to feet at all. The following tableau shows how this pattern emerges (colors represent different strata):

(2)

input	winner	loser	*σF	*$F\sigma$	TROCH	PARSE	IAMB
3syll	$\breve{\sigma}\breve{\sigma}\breve{\sigma}$	$\breve{\sigma}(\acute{\sigma}\sigma)$	W			L	W
3syll	$\breve{\sigma}\breve{\sigma}\breve{\sigma}$	$(\sigma\acute{\sigma})\breve{\sigma}$		W		L	W
1syll	$\breve{\sigma}$	$(\acute{\sigma})$			W	L	W
2syll	$(\acute{\sigma}\sigma)$	$\breve{\sigma}\breve{\sigma}$				W	L

In odd-syllable forms, *$F\sigma$ and *σF cannot be satisfied by placement of binary feet only, but any attempt at insterting unary feet violates TROCH. In even-syllable forms, PARSE is best satisfied by full parsing of syllables to feet – anything less violates higher-ranked constraints. It can be demonstrated by way of mathematical proof that not only is this language not SL, it is not *star-free* (SF; McNaughton and Papert 1971). SF languages are also called *non-counting* languages, which more intuitively expresses why this sour grapes pattern is outside of the class – it relies on knowing if the string is of even or odd length. The SF class of formal languages is the next class of languages below the regular class, and so this sour grapes language's exclusion from the SF class demonstrates that it is fully regular. However, studies of natural language stress patterns as formal languages have shown them to overwhelmingly be SF (Rogers et al. 2013).

4. Discussion This paper has shown that a set of markedness constraints defined as SL in parallel OT can generate patterns that are provably fully regular. The specific pattern discussed is a novel sour grapes-like pattern for stress. This jump from SL to regular shows that limiting the CDL for constraints to some level of complexity is no guarantee of a typology of the same level of complexity. Further work will investigate this property of OT for other CDLs. Given a CDL of some complexity, of what complexity is its typology?

Selected References Eisner, J. (1997) *What constraints should OT allow?* – Jardine, A. and Heinz, J. (2016) *Markedness constraints are negative: an autosegmental constraintdefinition language* – McNaughton, R. and Papert, S. (1971) *Counter-free automata* – Prince, A. and Smolensky, P. (1993) *Optimality Theory* – Rogers et al. (2013) *Cognitive and subregular complexity*

Rhythmic Syncope in Subregular Phonology

Dustin Bowers, University of Arizona • **Yiding Hao**, Yale University

Rhythmic syncope is a phonological process in which every second vowel of a word is deleted. (1) shows an example of rhythmic syncope in 1930s Ojibwe (Bowers, To appear).

(1) /gʊtɪgʊmɪnʌgɪbɪnɑːd/ → [gtɪgmɪŋɪbnɑːd] "if he rolls him"

This talk presents a formal-language-theoretic analysis of rhythmic syncope, situating it within Chandlee's (2014) program to define strict subclasses of the finite-state functions, known as *subregular classes*, that attempt to capture the range of possible phonological transformations. We show that rhythmic syncope does not belong to the *tier-based input–output strictly local* functions (TIOSL, Chandlee et al., In prep), the largest subregular class proposed thus far. Based on this result, we define a new subregular class of functions, the *time-aligned tier-based strictly local* functions (TATSL), that naturally captures rhythmic syncope. However, we also present empirical evidence due to Bowers (To appear) suggesting that child learners may not be able to acquire rhythmic syncope. These empirical results suggest that the language acquisition process is not optimized for learning patterns not described by TIOSL functions.

1 Non-Strict-Locality of Rhythmic Syncope

The *tier-based input strictly local* (TISL) and *tier-based output strictly local* (TOSL) functions were proposed by Chandlee (2014) and Chandlee et al. (2015), respectively, as an extension of the *tier-based strictly local* (TSL) languages of Heinz et al. (2011) to functions between strings. The TIOSL functions generalize the TISL and TOSL functions. Let Σ be an alphabet of phonological symbols. For a number $k \geq 1$ and set $T \subseteq \Sigma$, an *input–output k-strictly local function on tier T* is a function $f : \Sigma^* \to \Sigma^*$ produced by scanning an input string one symbol at a time from left to right and producing a contiguous portion of the output at each time step. At each time step, the output produced may only depend on the current input symbol; the $k-1$ previous input symbols, ignoring symbols of T; and the $k-1$ previous output symbols, ignoring symbols of T.

Informally, our argument that rhythmic syncope is not TIOSL is as follows. Without loss of generality, assume that rhythmic syncope deletes vowels in even-numbered positions; i.e., starting from the second vowel. Let a be any vowel, and consider a long input string $/a^n/$, where $n \geq 4k$. After the first $2k$-many as have been scanned, rhythmic syncope has deleted k of these as, resulting in the partial output $[a^k]$. During the $(2k+1)$st time step, the current input symbol is a, and the $k-1$ previous input and output symbols, ignoring symbols of T, are a^{k-1} if $a \in T$ and $\varnothing$ otherwise. Because the current symbol is a vowel in an odd-numbered position, it is not deleted. During the next time step, the current input symbol is still a, and the $k-1$ previous input and output symbols, ignoring symbols of T, are still a^{k-1} if $a \in T$ and $\varnothing$ otherwise. However, because the current symbol is a vowel in an even-numbered position, it *is* deleted. This means that the behavior of rhythmic syncope is not determined by the current symbol and the $k-1$ previous input and output symbols, so it is not TIOSL.

2 Time-Aligned Tier-Based Strictly Local Functions

Intuitively, rhythmic syncope can be implemented by scanning the input from left to right if the incremental outputs can depend on the most recent *action* of the algorithm. If the most recent vowel is deleted, then the current vowel is not, and *vice versa.* This intuition forms the basis of the existing parallel (Kager, 1997; Blumenfeld, 2006) and serial (McCarthy, 2008) treatments of rhythmic syncope in Optimality Theory. Based on this idea, we represent actions as ordered pairs in $\Sigma \times \Sigma^*$.[1] Roughly speaking, a function produced by scanning the input and incrementally producing output is *time-aligned k-strictly local on tier T* if the output depends on the current input symbol and the $k-1$ most recent actions of the procedure.

3 Empirical Evidence of Unlearnability

Rhodes (1985) reports that Ojibwe lost rhythmic syncope after the 1930s, so that /gtɪgmɪŋɪbnɑːd/ is now the underlying form of the verb in (1). Bowers (To appear) confirms Rhodes's description through a series of surveys, and demonstrates that modern Ojibwe speakers growing up during the 1930s are aware of the 1930s forms. These results imply that the loss of alternating vowel deletion is not due to lack of exposure, suggesting that L1 language acquisition may be rationalistically biased against rhythmic syncope. Similar phenomena have been reported in Old Russian (Isačenko, 1970) and Old Irish (McManus, 1983).

4 Conclusion

We have shown that rhythmic syncope is not TIOSL, and we have proposed the TATSL functions as a further generalization of the TIOSL functions that extends the notion of strict locality to include dependencies between actions performed during a derivation. While our proposed class expands empirical coverage, the results of Bowers (To appear) suggest that the TATSL functions may be too powerful, yielding an interesting case of an empirical upper bound on the expressive power of phonology.

References

Blumenfeld, Lev A. 2006. *Constraints on Phonological Interactions*. Stanford, CA: Stanford University PhD Dissertation.

Bowers, Dustin. To appear. The Nishnaabemwin Restructuring Controversy: New Empirical Evidence. *Phonology* .

Chandlee, Jane. 2014. *Strictly Local Phonological Processes*. Newark, DE: University of Delaware PhD Dissertation.

Chandlee, Jane, Rémi Eyraud & Jeffrey Heinz. 2015. Output Strictly Local Functions. In Marco Kuhlmann, Makoto Kanazawa & Gregory M. Kobele (eds.), *Proceedings of the 14th Meeting on the Mathematics of Language*, 112–125. Chicago, IL: Association for Computational Linguistics. doi:10.3115/v1/W15-2310.

Chandlee, Jane, Rémi Eyraud & Jeffrey Heinz. In prep. Input–output strictly local functions and their efficient learnability.

Heinz, Jeffrey, Chetan Rawal & Herbert G. Tanner. 2011. Tier-based Strictly Local Constraints for Phonology. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 58–64. Portland, OR: Association for Computational Linguistics.

Isačenko, Alexander. 1970. East Slavic morphophonemics and the treatment of the jers in Russian: A revision of Havlík's law. *International Journal of Slavic Linguistics and Poetics* 13. 73–124.

Kager, René. 1997. Rhythmic vowel deletion in Optimality Theory. In Iggy Roca (ed.), *Derivations and Constraints in Phonology*, 463–499. Oxford, United Kingdom: Clarendon Press.

McCarthy, John J. 2008. The serial interaction of stress and syncope. *Natural Language & Linguistic Theory* 26(3). 499–546. doi:10.1007/s11049-008-9051-3.

McManus, Damian. 1983. A Chronology of the Latin Loan-Words in Early Irish. *Ériu* 34. 21–71.

Rhodes, Richard. 1985. Lexicography and Ojibwa Vowel Deletion. *Canadian Journal of Linguistics/Revue canadienne de linguistique* 30(4). 453–471. doi:10.1017/S0008413100011245.

[1]For example, $\langle a, b\rangle$ means "change a to b," $\langle a, \varnothing\rangle$ means "delete a," and $\langle a, ab\rangle$ means "epenthesize a b after a."

Diagnosing movement via the absence of c-command relations

Thomas Graf (Stony Brook U) & Sabine Laszakovits (UConn)

Keywords: c-command, computational linguistics, empirical generalization, islands, Minimalist Grammar, movement

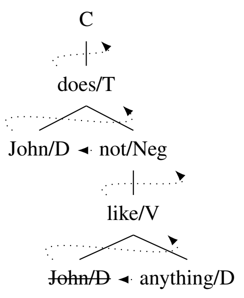
Synopsis. Graf and Shafiei (2019) propose a string-based computational mechanism for modeling c-command dependencies. We show that this account cannot handle dependencies that involve elements in distinct subtrees, as in parasitic gaps and the Coordinate Structure Constraint. However, all problematic cases involve movement. We thus propose a *computationally motivated diagnostic for distinguishing movement and c-command dependencies*:

(1) If a dependency applies across independent subtrees, then it involves movement (but not necessarily the other way round).

Formalizing c-command dependencies. Graf and Shafiei (2019) reduce c-command dependencies to constraints over a particular kind of string. Every node is associated with a *c[ommand]-string*, which encodes a right-to-left, bottom-up traversal of the sentence's dependency tree. This is indicated by arrows in (3), and (2) shows the resulting c-string of the direct object.

(2) C ↑ does ↑ John ← not ↑ like ↑ ~~John~~ ← anything

(3)



With c-strings, constraints like "NPIs must be c-commanded by negation" reduce to "the c-string of an NPI must contain negation". The primary interest of Graf and Shafiei (2019) lies in the computational complexity of such c-string constraints, which they argue is comparable to what is found in phonology (Graf and Mayer, 2018; Mayer and Major, 2018) and morphology (Aksënova, Graf, and Moradi, 2016). The observed computational parallelism between syntax on the one hand and phonology and morphology on the other hand suggests that c-strings enjoy some degree of cognitive reality.

Not all dependencies involve c-command. Graf and Shafiei's (2019) system cannot handle all syntactic dependencies. C-strings only keep track of c-commanders. They are too limited a representation to capture constraints that hold between elements that do not stand in a c-command relation. Two concrete examples are Across-the-Board movement exceptions to the Coordinate Structure Constraint and the licensing of parasitic gaps.

In **ATB-extraction** (Ross, 1967; Williams, 1978; i.a.), extraction out of a conjunct is possible only if the same phrase is extracted from all conjuncts.

(4) I wonder who [John introduced [the father of { _ / *Jill}] to this girl] and [Bill introduced [the mother of _] to this boy].

The c-string account necessarily fails here because the differences in structure do not cause a difference in c-strings. In both sentences, the second gap has the c-string in (5):

(5) C ↑ T ↑ I ← wonder ↑ ~~I~~ ← C[+wh] ↑ who ← and ↑ -ed ← -ed ↑ Bill ← introduce ↑ ~~Bill~~ ← the ↑ mother ↑ of ↑ ~~who~~

Since (4) with two gaps is well-formed, this c-string cannot be illicit. But every other c-string in (4) can also occur in a well-formed sentence. Hence c-strings cannot capture the ill-formedness of (4) with a single gap.

Parasitic gaps need to "piggy-back" on a mover in order to be licensed. In (6) the gap in the adjunct clause is licensed only if the matrix object undergoes wh-movement.

(6) a. What did John [review a copy of _] [before Mary read _]?
b. *What did John [eat a sandwich] [before Mary read _]?
c. *Who [reviewed a copy of this book$_1$] [before Mary read _$_1$]?

A c-string constraint for the paradigm in (6) needs to capture the dependency of the adjunct gap on the matrix object. However, no c-string in (6) can contain both the adjunct object and the matrix object. The c-string for the adjunct gap is given in (7):

(7) did ↑ what ← ~~did~~ ↑ before ↑ T[past] ↑ Mary ← read ↑ ~~Mary~~ ← ~~what~~

Generalization. The examples above involve movement in the sense that they either limit it or absolutely require it for licensing. This is in contrast to c-command dependencies such as binding, morphosyntactic case, and NPI-licensing, which may interact with movement but do not limit or require it as an integral part of the licensing. Curiously, we have been unable to find any example of the latter category that cannot be expressed with constraints on c-strings. We thus put forward the following conjecture:

(8) Every constraint that cannot be captured with c-strings must involve movement.

Note that the other direction does not hold: some constraints on movement can be described by c-command.

C-string constraints on movement. Some syntactic constraints involving movement can be fully described by making reference to c-command. **Adjunct islands** (Ross, 1967) ban movement out of adjuncts such as *because*-clauses, (9).

(9) *What did you fall asleep [because John was reading _]?

In **freezing effects**, an XP cannot move out of a YP that has already moved. This is illustrated in (10) for the Subject Condition, assuming movement of the subject according to the VP-Internal Subject Hypothesis.

(10) *?I wonder [who [friends of _] hired Mary].

Each constraint corresponds to a restriction on c-strings: (11a) for the Adjunct Island Constraint; (11b) for Freezing.

(11) a. *... XP ... *because* ↑ ... XP b. *... XP ... YP ... YP ↑ ... XP

Implications. To the extent that our generalization in (1)/(8) holds, we propose a *new diagnostic for movement*: Whenever a syntactic constraint cannot be formulated by making reference to the c-commanders and c-commandees of the element requiring licensing, there must be movement of this element or of the elements licensing it. That constraints on movement go beyond c-strings is not surprising. Previous work on Minimalist grammars has shown that movement must satisfy certain constraints in order to keep complexity in check (Salvati, 2011; Stabler, 1997). These constraints are computationally similar to ATB-extraction and parasitic gaps, and thus it cannot be captured by c-strings, either. The very essence of Move requires a different computational machinery, and this difference is also reflected in what constraints can apply to Move.

Selected references: Graf, T. & C. Mayer (2018): "Sanskrit n-Retroflexion is Input-Output Tier-Based Strictly Local", SIGMorPhon 15. • Graf, T. & N. Shafiei (2019): "C-command dependencies as TSL string constraints", SCiL 2.