Subregular Tree Transductions, Movement, Copies, Traces, and the Ban on Improper Movement

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

Abstract

Extending prior work in Graf (2018, 2020, 2022c), I show that movement is tier-based strictly local (TSL) even if one analyzes it as a transformation, i.e. a tree transduction from derivation trees to output trees. I define *input strictly local (ISL) tree-to-tree transductions with (lexical) TSL tests* as a tier-based extension of ISL tree-to-tree transductions. TSL tests allow us to attach each mover to all its landing sites. In general, this class of transductions fails to attach each mover to its final landing site to the exclusion of all its intermediate landing sites, which is crucial for producing output trees with the correct string yield. The problem is avoided, though, if syntax enforces a variant of the Ban on Improper Movement. Subregular complexity thus provides a novel motivation for core restrictions on movement while also shedding new light on the choice between copies and traces in syntax.

1 Introduction

Subregular syntax (Graf, 2018; Graf and De Santo, 2019) is a recent research program that explores whether syntactic dependencies, when modeled over suitable representations, fall within very restricted classes in the subregular hierarchy of formal (string or tree) languages. The program has many parallels to *subregular phonology* (see Heinz 2018 and references therein), which has shown that phonology is very restricted in its expressivity: I) well-formedness conditions in phonology are *strictly local* (SL), *tier-based strictly local* (TSL) (Heinz et al., 2011; McMullin, 2016), or some natural extension of TSL (Graf and Mayer, 2018; Mayer and Major, 2018; De Santo and Graf, 2019), and II) a large number of phonological mappings from underlying representations to surface forms are *input strictly local* (ISL) (Chandlee, 2014; Chandlee and Heinz, 2018), with only some falling into more complex classes (Jardine, 2016; Heinz, 2018). The limited nature of phonology furnishes new learning algorithms and novel explanations of typological gaps, and subregular syntax seeks to replicate this success for syntax.

A lot of attention in subregular syntax has been devoted to the operations *Merge* and *Move* in Minimalist syntax and Minimalist grammars (Stabler, 1997, 2011). Merge establishes head-argument relations, whereas Move relates a subtree to multiple positions in the structure. Graf (2018) showed that the constraints that regulate the application of Merge and Move in the syntactic derivation are SL for Merge and TSL for Move, which mirrors the central role of these two classes in phonology. But Merge and Move are structure-building operations and thus inherently transductive: a syntactic derivation is translated into a specific output structure. Recently, the ISL string transductions from subregular phonology have been generalized to trees (Graf, 2020; Ji and Heinz, 2020; Ikawa et al., 2020), and it is fairly easy to see that Merge can be construed as an ISL tree transduction.[1] However, ISL tree transductions cannot handle the long-distance dependencies induced by Move (the long-distance nature of Move is also why the constraints on Move are TSL but not SL). An upper complexity bound on Move exists in the form of deterministic multi bottom-up tree transductions (Kobele et al., 2007), but a tighter, subregular bound remains to be established.

This paper provides a subregular class of transductions for Move by enriching (deterministic) ISL tree-to-tree transductions with a specific TSL mech-

[1] The three generalizations in Graf (2020), Ji and Heinz (2020) and Ikawa et al. (2020) are all distinct and probably incomparable. Graf (2020) generalizes the context-based definition of ISL in Chandlee and Heinz (2018), Ji and Heinz (2020) takes as their vantage point the finite-state machine definition of ISL in Chandlee (2014), and Ikawa et al. (2020) starts with the logic-based perspective of ISL string transductions. Despite these differences, all three can handle the mapping from dependency trees to phrase structure trees *modulo* movement. For the rest of the paper, I will use the term *ISL tree transductions* to refer to the specific version defined in Graf (2020).

Proceedings of the Society for Computation in Linguistics (SCiL) 2023, pages 289-299.
Amherst, Massachusetts, June 15-17, 2023

anism that makes it possible to attach movers to their landing sites. This is sufficient to implement a copy-based version of movement, which is commonly assumed in Minimalist syntax. Producing a structure with the correct string yield, however, requires the ability to distinguish final landing sites from intermediate ones so that movers can be attached only to the former while the latter are filled with traces. The extended version of ISL tree transductions in this paper cannot draw this distinction in the general case, but it is possible in the special case where the distinction is lexically inferrable (in subregular terms, it is SL-1): given a mover m with a set $S := \{f_1, \ldots, f_n\}$ of features that tell us which movement steps m undergoes, inspection of S is sufficient to determine which f_i is the final movement step. This is a relaxed variant of the Ban on Improper Movement (BoIM), and I conjecture that this *output-oriented BoIM* is satisfied in all natural languages.

The paper proceeds as follows. The background section in §2 starts with a general overview of the assumed syntactic formalism, in particular feature-annotated lexical items, dependency trees, and tree tiers (§2.1). This is followed in §2.2 by a discussion of the ISL tree-to-tree mappings in Graf (2020), which are then extended with lexical TSL tests in §3 to capture basic cases of movement. As we will see in §4, this is sufficient to attach movers to all their landing sites. But correct linearization requires placing each mover only in its final landing site, which is a harder problem and prompts my conjecture that all languages satisfy the output-oriented BoIM. A few remaining issues with this overall system are discussed in §5. While care has been taken to make the paper as approachable as possible, it necessarily presupposes a certain amount of familiarity with subregular linguistics, in particular subregular syntax. The reader may want to consult Graf (2022a,b) for a less technical introduction.

2 Background

2.1 Features, dependency trees, and tiers

Subregular syntax measures the complexity of syntax not over strings but over specific types of tree representations. Following Graf and Kostyszyn (2021) and Graf (2022c), I take syntactic derivations to be encoded in the form of dependency trees where each node is a feature-annotated lexical item (LI) in the spirit of Minimalist grammars (Stabler, 1997, 2011).

Definition 1 (Lexical item). Every lexical item is a member of $\Sigma \times \text{Sel}^* \times \text{Lcr}^* \times \text{Cat} \times \wp(\text{Lce})$, where Σ is the set of *phonetic exponents*, Sel is the set of *selector features* $\texttt{F}^+$, Lcr is the set of *licensor features* $\texttt{f}^+$, Cat is the set of *category features* $\texttt{F}^-$, and Lce is the set of *licensee features* $\texttt{f}^-$. ⌟

Category and selector features (by convention in upper case) regulate the application of Merge to establish head-argument relations. Licensor and licensee features (in lower case) trigger Move, with licensor features appearing on the target of movement while licensee features mark the phrase that is moving. The order of features on an LI indicates the order of the operations in which it participates. In contrast to standard MGs, licensee features are unordered so that a mover with licensee features $\texttt{f}_1^-, \ldots, \texttt{f}_n^-$ targets, for each $\texttt{f}_i^-$, the closest properly dominating node with $\texttt{f}_i^+$ ($1 \leq i \leq n$). The removal of order for licensee features does not affect weak generative capacity — this is an easy corollary of the single movement normal form theorem for MGs (Graf et al., 2016).[2] To reduce clutter, we omit $\{\}$ for LIs with no licensee features. In line with MG convention, I use a double colon to separate the LI's phonetic exponent from its feature annotation.

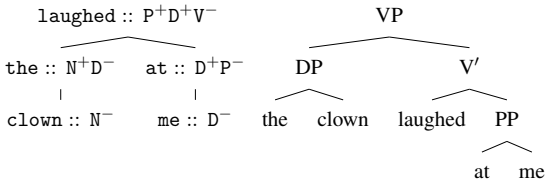
Example. The noun *movie* corresponds to the LI $\texttt{movie} :: \texttt{N}^-$ with phonetic exponent *movie* and category feature $\texttt{N}^-$. The empty T-head — commonly assumed in Minimalist syntax as furnishing the surface position for subjects — is $\varepsilon :: \texttt{V}^+\texttt{nom}^+\texttt{T}^-$. This means that after selecting a VP, the empty T-head provides a landing site for subject movement via $\texttt{nom}^+$, at which point it becomes a full TP that can be taken as an argument by another LI. The LI $'\texttt{s} :: \texttt{N}^+\texttt{D}^+\texttt{D}^- \{\texttt{nom}^-, \texttt{wh}^-\}$ is a possessive marker that takes an NP as its complement, a DP as its specifier, is then selected by another LI with $\texttt{D}^+$, and finally undergoes two movement steps: subject movement via $\texttt{nom}^-$, and wh-movement via $\texttt{wh}^-$. The order of the two movement steps is not fixed and depends on whether the closest properly dominating LI with a matching licensor feature carries $\texttt{nom}^+$ or $\texttt{wh}^+$.

Definition 2 (Dependency tree). Let Lex be a fi-

[2] The definition of LIs above also differs from that of standard MGs in that it does not allow any licensor features to appear before any selector features. This is just a matter of convenience and nothing in this paper hinges on this additional restriction.

nite set of LIs, and $\text{Lex}^{(i)} \subseteq \text{Lex}$ the set of all LIs in Lex with i selector features. The set $\mathbb{D}$ of *(freely combined) dependency trees over* Lex is defined recursively: $l \in \mathbb{D}$ for all $l \in \text{Lex}^{(0)}$, and for all $d_1, \ldots, d_n \in \mathbb{D}$ and $l \in \text{Lex}^{(n)}$, $l(d_n, \ldots, d_1) \in \mathbb{D}$. If m is the mother of node n and n has exactly i right siblings, we say that n is the $(i+1)$-th argument of m. ⌟

Example. A dependency tree for a simple VP is shown below with its corresponding bare phrase structure tree. Each mother-daughter relation in the dependency tree encodes a head-argument relation established via application of Merge.



In general, dependency trees have to satisfy additional linguistic conditions. The root must carry category feature $\texttt{C}^-$, and if m's i-th selector feature is $\texttt{F}^+$, then its i-th argument must carry category feature $\texttt{F}^-$. These constraints regulate the application of Merge and are of little interest for the purposes of this paper. The constrains on Move, on the other hand, merit detailed discussion as they illustrate the use of *tree tiers*.

Definition 3 (Tiers). Let $d \in \mathbb{D}$ be a dependency tree over Lex, and let $T \subseteq \text{Lex}$ be a *tier alphabet*. Given a node x, the predicate $T(x)$ is true iff x is an LI in T. The *T-tier* of d is defined in terms of T-dominance ($\lhd_T^+$), T-mother-of ($\lhd_T$), and T-left-sibling ($\prec_T$), which in turn are expressed in terms of proper dominance in d ($\lhd^+$), reflexive dominance in d ($\lhd^*$), and the left sibling relation in d ($\prec$).

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

In order to ensure that every tier is a tree, we stipulate that there is a unique node $\rtimes$ such that every node on tier T is either identical to $\rtimes$ or is T-dominated by $\rtimes$. We also stipulate that each leaf is the mother of a distinguished element $\ltimes$. ⌟

Example. The tier alphabet $\texttt{nom}$ of the $\texttt{nom}$-tier contains all LIs with $\texttt{nom}^-$ or $\texttt{nom}^+$, and nothing else. Similarly, the tier alphabet $\texttt{wh}$ of the $\texttt{wh}$-tier contains all and only those LIs that carry $\texttt{wh}^-$ or $\texttt{wh}^+$. The corresponding tier mother-of relations $\lhd_{\texttt{nom}}$ and $\lhd_{\texttt{wh}}$ are shown in Fig. 1 with dashed and dotted lines, respectively, for the dependency tree for *Who said that the clown laughed at me*. As shown in the same figure, these tiers can also be depicted as separate *projections* of the dependency tree.

Intuitively, tiers capture a specific kind of relativized locality (related to but distinct from Rizzi's (1990) notion of Relativized Minimality). If x is the T-mother of y, then x is the closest node that properly dominates y and belongs to a fixed subset T of Lex. For movement, each tier factors out all LIs that are not pertinent to that type of movement. In order for a dependency tree to be well-formed, the following two conditions must hold for every $\texttt{f}$-tier, where $\texttt{f}$ is some movement type ($\texttt{nom}$, $\texttt{wh}$, and so on): I) if x carries $\texttt{f}^-$, then its tier mother carries $\texttt{f}^+$, and II) if x carries $\texttt{f}^+$, exactly one of its tier daughters carries $\texttt{f}^-$.

Mathematically, these conditions are expressed for each tier T via a *licensing function* f_T that maps every $l \in T$ to a string language over T. Tier T is well-formed iff it holds for every node n of T with label l and tier daughters $d_1, \ldots, d_n$ that $d_1 \cdots d_n$ is a string in $f_T(l)$.[3] For example, if l is an LI with $\texttt{f}^+$, then $f_T(l)$ is the set of all strings over T that contain exactly one LI with $\texttt{f}^-$. That every LI with $\texttt{f}^-$ has a tier mother with $\texttt{f}^+$ follows indirectly from the fact that only LIs with $\texttt{f}^+$ may have LIs with $\texttt{f}^-$ in their daughter string.

The complexity of the conditions on Move is measured in terms of the complexity of the string languages used in the licensing functions. A constraint C on a set D of dependency trees over Lex is in the class TSL[TSL] (where TSL is short for *tier-based strictly local*) iff there is some $T \subseteq \text{Lex}$ such that I) f_T maps every $l \in T$ to a TSL-string language in the sense of Heinz et al. 2011 ("the daughter strings are TSL"), and II) for every $d \in D$, C is satisfied in d iff the T-tier of d is well-formed ("C is local over tree tiers"). The two constraints above on movement are TSL[TSL] in this sense (see Graf and Kostyszyn, 2021).

[3] The use of a string-based licensing function is necessary because tree tiers are unranked. There is no upper bound on how many daughters may have, and hence the licensing relations between a mother and its daughters has to be modeled as a licensing relation between a mother and its string of daughters.

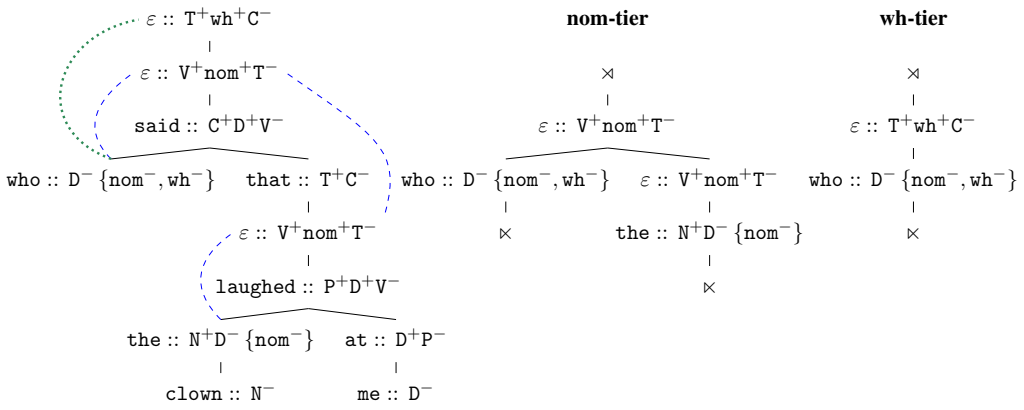


Figure 1: Left: dependency tree for *who said that the clown laughed at me*, with dashed lines representing $\lhd_{\text{nom}}$ and dotted lines representing $\lhd_{\text{wh}}$; Middle and Right: corresponding depictions as tree tiers

2.2 ISL tree-to-tree mappings

With our syntactic representations and the notion of tree tiers firmly in place, it only remains for us to define deterministic *input strictly local* (ISL) tree-to-tree transductions before we start our investigation of movement as a subregular transduction in §3.

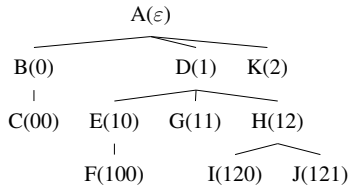
Deterministic ISL transductions, also called *ISL mappings*, were first defined in subregular phonology for the string-to-string case (Chandlee, 2014, 2017; Chandlee and Heinz, 2018). The ISL string-to-string mappings were subsequently generalized to (non-deterministic) tree-to-tree transductions in Graf (2020). An ISL tree transduction τ is specified by a finite number of rewrite rules. The left-hand side consists of a tree with one distinguished node h that is to be rewritten — the rest of the tree just provides the strictly local context in which this specific rule must be applied to h. The right-hand side consists of a tree with indexed *ports* $\square_1$, $\square_2$, …, $\square_n$ ($n \geq 0$) such that each $\square_i$ is filled with the output of τ for the i-th daughter of h. Figure 2 gives a simple example for mapping a dependency tree without movement (and with at most two arguments per LI) to its corresponding bare phrase structure tree — the reader is advised to study this example carefully before moving on to the formal definition.

We first put in place some common concepts from the tree transducer literature. A Σ-tree is a finite tree over alphabet Σ. We assume that all Σ-trees have a finitely bounded branching factor. Given a Σ-tree t, each node n in t is given a unique Gorn address $a(n)$ (Gorn, 1967): $a(n) = \varepsilon$ if n is the root of t, and otherwise $a(n) = ui$, where u is the Gorn address of the mother of n and i is the number of left siblings of n. A Σ-*tree context* c is the result of replacing $n \geq 1$ leaves in a Σ-tree with distinguished symbols drawn from a set of ports, which are denoted with $\square_i$, $i \geq 1$. Given such a context c and Σ-trees or Σ-tree contexts $t_1, \ldots, t_n$, $c\{1 : t_1, \ldots, n : t_n\}$ is the result of replacing $\square_i$ in c with t_i.

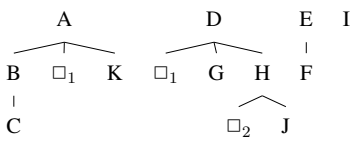
In order to determine the configurations in which ISL rewrite rules may apply, we introduce the notion of a tree disassembly.

Definition 4 (Tree dissassembly). A *disassembly* of tree t at addresses b, ba_1, …, ba_n is an $(n+2)$-tuple that consists of I) t with the subtree s at b replaced with $\square_1$, II) s with the subtrees at addresses ba_1, …, ba_n replaced with $\square_1$, …, $\square_n$, and III) the subtrees at addresses ba_1, …, ba_n. ⌟

Example. Consider the tree t below, with each node followed by its Gorn address in parentheses.



The disassembly of t at addresses 1, 10, and 120 consists of the following trees/contexts:



Next we define what ISL rewrite rules may look like and how a given rule may apply within a tree.

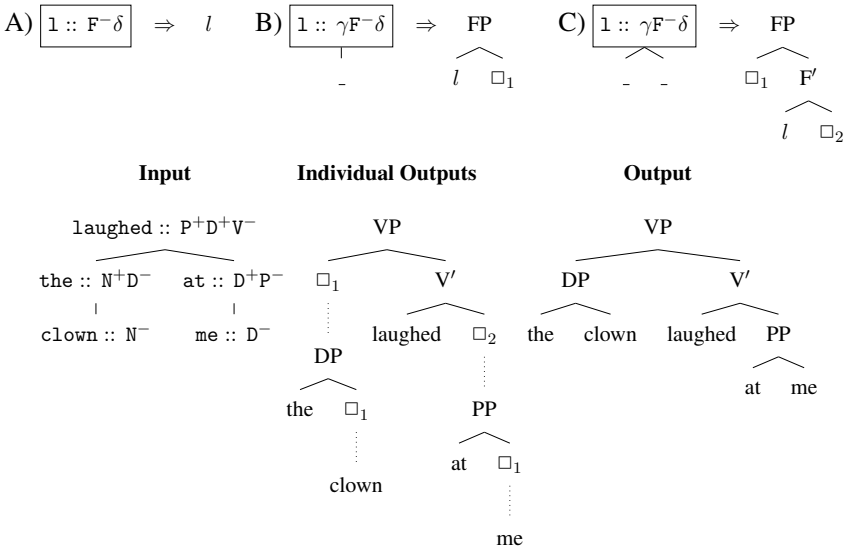
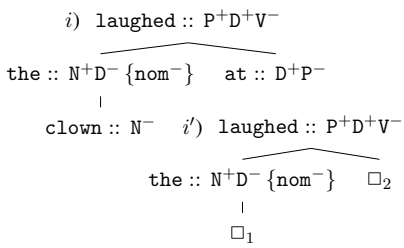


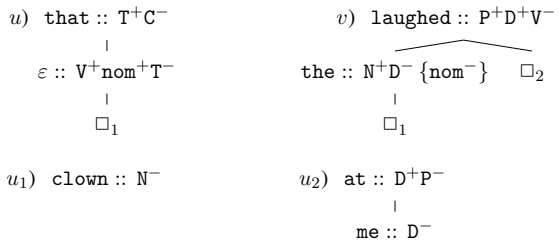
Figure 2: ISL rewrite rules for converting movement-free dependency trees to bare phrase structure trees (top) with example (bottom); boxes around nodes indicate which nodes should be rewritten, γ is a non-empty string of selector features, δ is a (possibly empty) set of licensee features, and _ matches any node

Definition 5 (ISL rewrite rule). An *ISL rewrite rule* is a triple $r := \langle i, a, o \rangle$ where the *input environment* i is a Σ-tree, a is the Gorn address of some node in i, and the *output context* o is a Σ-tree context. Suppose w.l.o.g. that i has exactly n leaf nodes at addresses $a_1, \ldots, a_n$ ($n \geq 1$) and let i' be the result of replacing each node at address a_j with $\Box_j$ ($1 \leq j \leq n$). Then r *matches* tree t at address b iff t has a disassembly $\langle u, i', u_1, \ldots u_n \rangle$ at addresses $b, ba_1, \ldots, ba_n$ such that both of the following hold: I) $t = u\{1 : i'\{1 : u_1, \ldots, n : u_n\}\}$, and II) for $1 \leq j \leq n$, the node at address a_j in i has the same label as the node at address ba_j in t. A node at address ba in t can be rewritten by r iff r matches t at address b. ⌟

Example. Consider a rewrite rule $r := \langle i, a, o \rangle$, with $a = 0$ and i as shown below (together with its counterpart i'):



Note that the ports of i' have addresses $a_1 := 00$ and $a_2 := 1$. We show that i matches the dependency tree t in Fig. 1 at address $b = 00100$. We first disassemble t at addresses b, $ba_1 = 0010000$, and $ba_2 = 001001$. This yields four trees/contexts u, v, u_1, u_2. For space reason, we only show the subtree of u rooted in that :: T⁺C⁻.



As v is identical to i', it holds that $t = u\{1 : v\{1 : u_1, \ldots, n : u_n\}\} = u\{1 : i'\{1 : u_1, \ldots, n : u_n\}\}$. It is also the case that the nodes of i at addresses a_1 and a_2 have the same labels as the nodes in t at addresses $ba_1 = 0010000$ and $ba_2 = 001001$. Taken together, this means that r matches t at address b. Consequently, r can rewrite as o the node at address $ba = 001000$ in t, which is the :: N⁺D⁻. Note that if the root of i had a third daughter labeled, say, *maliciously*, i would no longer match t at any address.

Definition 6 (Deterministic ISL transduction). Given a set R of ISL rewrite rules, we say that R is *deterministic* iff there are no two rewrite rules $\langle i_1, a_1, o_1 \rangle$ and $\langle i_2, a_2, o_2 \rangle$ in R such that $o_1 \neq o_2$ and there exists a Σ-tree t and node n of t such that n can be rewritten by both rewrite rules.

For each deterministic set R of ISL rewrite rules,

$R(t, n)$ denotes the unique output context o for node n in tree t (if no such o exists, $R(t, n)$ is undefined). We extend this to t in a recursive fashion: if t contains only node n, then $R(t) := R(t, n)$, and if $t := n(s_1, \ldots, s_z)$ (each s_i a Σ-tree), then $R(t) := R(t, m)\{1 : R(t, d_1), \ldots, z : R(t, d_z)\}$. A tree-to-tree transduction τ with domain D is *deterministic input strictly local* iff there is a finite deterministic set R of ISL rewrite rules such that $\tau(t) = R(t)$ for all $t \in D$. In this case, we also call τ an *ISL (tree-to-tree) mapping*. ⌟

3 Movement as a subregular transduction

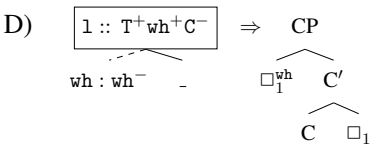
Move cannot be captured with ISL tree-to-tree mappings. The problem is not with the determinism of those mappings. In the formalism used in this paper, Move is a deterministic operation in the sense that the landing sites of a mover can be inferred deterministically from LIs' feature annotations (and as a result the definition of ISL mappings in this paper can safely avoid many complexities in the definitions of non-deterministic ISL transductions in Graf 2020). But while movement is deterministic, it is also unbounded — a mover and its target site can be arbitrarily far apart. Since ISL transductions must be definable in terms of a finite set of rewrite rules, and since each rewrite rule $\langle i, a, o\rangle$ is limited to the finite structural context given by i, ISL transductions cannot handle such unbounded dependencies. For example, we may want to rewrite a node n that carries $\texttt{wh}^+$ as a phrase whose specifier is filled by a wh-mover, but our rewrite rules provide no means to refer to this mover unless it happens to be very close to n. In order to capture movement, ISL rewrite rules must be able to refer to nodes that can be arbitrarily far away.

Tiers provide a natural solution to this problem. We already saw in §2.1 that tiers play a key role in movement — even though movement is unbounded over dependency trees, it is local over tiers. All we have to do is to incorporate this tier-based locality into ISL transductions.

Suppose, then, that we enrich our rewrite rules with another type of ports, called *tier ports*. If we are to rewrite a node n that is part of some `f`-tier, then its output context can include `f`-tier ports. The left-hand side of rewrite rules now also specify a specific test, and a tier port can only pick out the node that passes this test (the node must be unique!). The use of tier tests in the rewrite rules is why I call this new class of transductions *ISL tree-to-tree mappings with TSL tests*.

In this paper, the TSL tests are particularly simple as each one corresponds to a fixed set of LIs that pass the test. Just like the licensing function of TSL in §2.1 could define string languages of various complexity levels all the way up to recursively enumerable, the tests for tier ports can be of arbitrary complexity. But at least for movement, the maximally restricted class of lexical tests (in subregular parlance, SL-1 tests) is sufficient. Hence this paper restricts itself to the even weaker subclass *ISL tree-to-tree mappings with lexical TSL tests*.

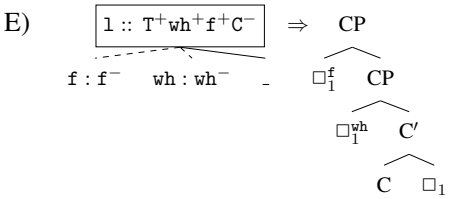
Let us consider how this system captures simple cases of movement. To this end, we add a new rewrite rule to the set in Fig. 2.



This rule targets C-heads that select a TP and provide a landing site for wh-movement. Every such C-head is rewritten as a CP where the complement is filled by the output of the first daughter in the dependency tree, whereas the specifier is filled by the output of the unique node x such that the C-head is the `wh`-tier mother of x and x carries $\texttt{wh}^-$. This is sufficient to connect movers to their landing sites.

Rule D uses two new notational devices: dashed lines for the tier mother-of relation, and tier ports. The dashed line in D leads to a special node that starts with the name of a tier (`wh` in this case), followed by a colon, and the set of LIs on this tier that can be picked out by the tier port $\square_1^{\texttt{wh}}$. Here $\texttt{wh}^-$ is used as a shorthand for the set of all LIs that carry $\texttt{wh}^-$. The tier port $\square_1^{\texttt{wh}}$ is to be filled with the output of the unique node that is a `wh`-tier daughter of the node to be rewritten and carries $\texttt{wh}^-$.

In a more elaborate case where the C-head also attracts some other kind of `f`-mover, the rule would look as follows.



A fully worked out example is shown in Fig. 3 for the sentence *who said that*, where the subject *who*

first undergoes subject movement to Spec,TP and then wh-moves to Spec,CP.

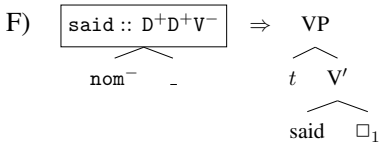
Quite generally, adding lexical TSL tests to ISL tree-to-tree mappings only requires three minor tweaks. First, each rewrite rule is extended to also include a finite (and possibly empty) collection of TSL tests. Second, the notion of a rewrite rule *matching* a tree at a given address b is expanded to also require partial tier matches: if the rule specifies that the node at address a is an $\texttt{f}$-tier mother of an element that passes some test ϕ, then the node at address ba in the dependency tree must be part of the $\texttt{f}$-tier and must have exactly one node x among its $\texttt{f}$-tier daughters such that x passes test ϕ. Finally, the definition of $R(t)$ is amended to include substitution into tier ports. The full definition that incorporates all these changes is given in the appendix.

Inspection of the example in Fig. 3 quickly reveals that the solution laid out above does not quite work as expected for movement. It attaches every mover to all its landing sites, and as a result the bare phrase structure tree contains multiple instances of *who*. In other words, the rewrite rules above implement a copy-theory of movement, but they do not capture the fact that moved phrases are only pronounced in their final landing site. A solution is readily available, though, provided one can tell the final movement step of a mover just from its feature make-up.

4 Linearization and the output-oriented BoIM

Our previous solution for movement runs into problems because movement actually consists of two steps: attaching the mover to all its landing sites, and delinking it from all positions that are not its final landing site.

Delinking itself is fairly simple from the perspective of ISL transductions. Consider the example below for delinking the moving *who* in Fig. 3 from its base position under *said*.



Here $\texttt{nom}^-$ is a shorthand for any LI carrying $\texttt{nom}^-$. The rewrite rule thus replaces the left daughter with a trace provided it undergoes subject movement. Note that since we only care about well-formed dependency trees where every licensee feature has a matching licensor feature on some other node, the fact that the left daughter carries $\texttt{nom}^-$ guarantees that it will undergo subject movement and hence should not be linearized as an argument of the verb. The feature make-up of the LI thus determines whether its base position should be replaced with a trace.

Things are trickier, though, when we consider intermediate landing sites such as Spec,TP for *who*. Since licensee features are not ordered, we cannot tell whether $\texttt{who} :: \texttt{D}^- \{\texttt{nom}^-, \texttt{wh}^-\}$ first undergoes nom-movement or wh-movement. The assumption that licensee features are unordered is crucial for the tier-based perspective of movement, it cannot be easily done away with. It seems, then, that our delinking trick for base positions does not carry over to intermediate landing sites like Spec,TP. We cannot tell from the local context of the T-head whether the subject mover with $\texttt{nom}^-$ will move on to a higher position via wh-movement, or if it has already done so and will thus stop in Spec,TP. One may be tempted to try ideas like merging the nom-tier and the wh-tier into a single tier, but these do not work either because then a mover and its landing site may no longer stand in a mother-daughter configuration. While a mathematical proof is still outstanding, it seems that there is no way in the current system to correctly distinguish final from intermediate landing sites.

Linguists will point out, though, that Spec,TP cannot be the final landing site for *who* due to the *Ban on Improper Movement* (BoIM): once a mover undergoes an instance of A′-movement like wh-movement, it can no longer undergo any A-movement steps such as subject movement. The BoIM rules out sentences like the illicit *who wonders [t John saw t]*, where *who* first wh-moves to Spec,CP of the embedded clause before undergoing subject movement into the matrix clause.

In light of the BoIM, it is readily apparent from the feature make-up of $\texttt{who} :: \texttt{D}^- \{\texttt{nom}^-, \texttt{wh}^-\}$ that it first undergoes nom-movement and then wh-movement. Consequently, the purely feature conditioned delinking strategy still works and one could something like rule G below for rewriting the T-head. Rule H for rewriting the C-head looks almost exactly the same except that we insert the mover and not a trace. In both rules, $\{\texttt{nom}^-, \texttt{wh}^-\}$ matches every LI that carries at least those two licensee features.

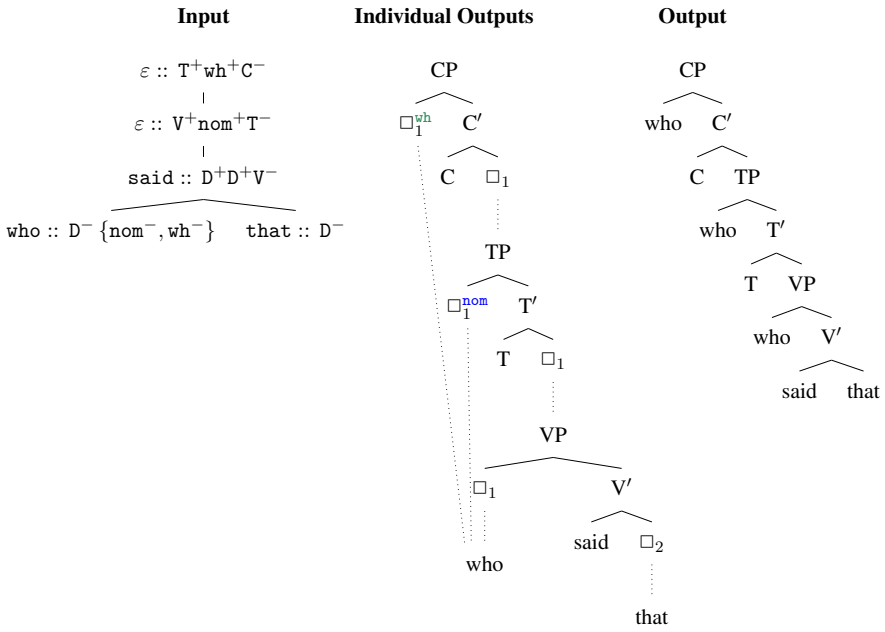
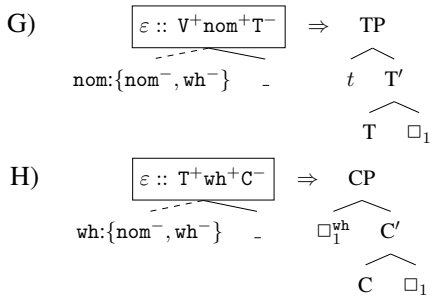


Figure 3: The dependency tree for *who said that* is rewritten into the corresponding bare phrase structure tree.



At least in the case of subject movement and wh-movement, then, ISL tree-to-tree transductions with TSL tests allow us not only to associate a mover with all its landing sites, but also to produce linearized output structures with the correct string yield.

In order for this solution to extend to all of syntax, however, a stronger property has to be in place.

Definition 7 (Output-oriented BoIM). For no LI l with set $\{\mathtt{f}_1^-, \ldots, \mathtt{f}_n^-\}$ of licensee features may there be well-formed dependency trees t_1 and t_2 such that I) both t_1 and t_2 contain l, and II) l's final movement step is $\mathtt{f}_i$ in t_1 and $\mathtt{f}_j$ in t_2 ($i \neq j$). ⌟

In other words, for every LI l one can always predict its final movement step based purely on inspection of the LI itself.

I conjecture that the output-oriented BoIM is a universal property of movement across languages. This is prompted by two observations. First, a preliminary analysis of the MG corpus (Torr, 2017) suggests that the output-oriented BoIM holds for the trees in that treebank. In fact, the licensee features used in that corpus seem to obey an even stronger restriction: for every LI l that carries, say, $\mathtt{f}^-$ and $\mathtt{g}^-$, it is always the case that l undergoes $\mathtt{f}$-movement before $\mathtt{g}$-movement. While corpora represent just a finite slice of a possibly infinite range of licit configurations, it is encouraging that the conjecture clears this first hurdle with ease.

The second argument is more indirect: While the syntactic literature has noted potential exceptions to the BoIM, those do not directly carry over to the system used here. Consider the case of hyperraising in Zulu (see Zyman 2023 and references therein). Here a DP undergoes A-movement from a position in the embedded clause to some position in the matrix clause, yielding a configuration similar to the illicit English sentence *Mary seems [that will go home]*. Minimalists assume for independent reasons that *Mary*, rather than moving directly from the embedded subject position to the matrix subject position, has to stop in Spec,CP of the embedded clause. As the latter is an instance of A′-movement, hyperraising seems to involve an A′-movement step to Spec,CP followed by A-movement to the subject position of the matrix clause. But this A′-movement step is driven by theoretical considerations related to successive cyclic movement, which is treated very differently

in MGs and subregular syntax. The phenomena that are used to motivate successive cyclic movement, e.g. wh-agreement in Irish, can be captured without such movement in TSL syntax (Graf, 2022c). Without successive cyclic movement, though, hyperraising is no longer a counterexample to the standard BoIM, let alone the output-oriented BoIM that is needed in this system of ISL transductions with lexical TSL tests.

If the output-oriented BoIM turns out to be empirically robust, then the limits of ISL tree-to-tree transductions with TLS tests provide a novel motivation for the otherwise mysterious BoIM (which would then be a stronger implementation of the output-oriented BoIM). Subregular complexity might offer a computational third-factor explanation (Chomsky, 2005) for one of the most robust universals of syntax.

5 Remarks and open issues

The discussion so far has assumed that all movement steps are overt. Minimalist syntax and MGs both allow for covert movement steps, which do not affect linearization. In such systems, the final landing site of LI l with respect to linearization may be distinct from the landing site of its final movement step. This does not introduce any new challenges, though, as long as the following condition is met: for every set $S := \{\mathtt{f}_1^-, \ldots, \mathtt{f}_n^-\}$ of licensee features and every type of output structure (e.g. phrase structure tree, LF), one can tell directly from S whether $\mathtt{f_i}$-movement ($1 \leq i \leq n$) creates a copy or a trace at the landing site.

Another issue arises with successive cyclic movement. A common approach in MGs posits that successive cyclic movement is not feature-triggered but rather a result of the output mapping inserting traces and/or copies at specific positions along a movement path. ISL mappings with lexical TSL tests struggle with this because a node that is not on tier T cannot use T to test whether it is along a movement path. At the same time, putting, say, all C-heads on a tier T together with all wh-movers does not help either as the T-daughter of some C-head may then just be another C-head rather than the desired wh-mover. Instead of a transduction-based model of successive cyclic movement, one based on tier constraints may be more promising (cf. Graf, 2022c).

Finally, the complexity of copies vs. traces merits further exploration. Kracht (2001) observes that one can freely translate between copies and traces, but we saw that copy-based movement is simpler than trace-based movement because the latter requires additional restrictions on movement. Similarly, transductions with copying are more complex than linear transductions, yet the latter are sufficient for trace-based movement. This suggests that the subregular notions of complexity crosscut traditional ones in unexpected ways that may sometimes favor more complex machinery in one area in order to reduce complexity in another. These connections could only be hinted at in this paper but are ripe for future exploration from a mathematical perspective, e.g. in terms of DAG transductions (Drewes, 2017) as dependency trees with tier relations are essentially DAGs with labeled edges.

Conclusion

I have introduced (deterministic) ISL tree-to-tree transductions with TSL tests as a new class of subregular transductions that expands the ISL tree-to-tree transductions of Graf (2020) with the tier-based view of movement in Graf (2018, 2022c) in order to provide a subregular model of movement as a mapping from syntactic derivations (represented via dependency trees) to output structures. This class of transductions is still conceptually simple while offering enough expressivity to easily relate each mover to all its landing sites. The transductions in this class are too weak to distinguish final from intermediate landing sites, which is essential for obtaining the correct string yield from a syntactic derivation. However, it seems that a variant of the Ban on Improper Movement restricts syntax in just the right way to draw the necessary distinction between final and intermediate landing sites based purely on the feature make-up of the mover. It remains to be seen whether the output-oriented BoIM proposed here is indeed empirically viable, but the possibility is tantalizing as it promises a computational grounding for one of the best-known and most robust syntactic constraints.

Acknowledgments

This paper is dedicated to Christopher Graf, who entered this world a bit ahead of schedule, on the day of the submission deadline. I thank the reviewers for pushing this paper in a linguistically more comprehensive direction. The work reported in this paper was supported by the National Science Foundation under Grant No. BCS-1845344.

References

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

Jane Chandlee. 2017. Computational locality in morphological maps. *Morphology*, 27:599–641.

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

Noam Chomsky. 2005. Three factors in language design. *Linguistic Inquiry*, 36(1):1–22.

Aniello De Santo and Thomas Graf. 2019. Structure sensitive tier projection: Applications and formal properties. In *Formal Grammar*, pages 35–50, Berlin, Heidelberg. Springer.

Frank Drewes. 2017. On DAG languages and DAG transducers. *Bulletin of the European Association for Theoretical Computer Science*, 121.

Saul Gorn. 1967. Explicit definitions and linguistic dominoes. In *Systems and Computer Science, Proceedings of the Conference held at University of Western Ontario, 1965*, Toronto. University of Toronto Press.

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

Thomas Graf. 2020. Curbing feature coding: Strictly local feature assignment. In *Proceedings of the Society for Computation in Linguistics (SCiL) 2020*, pages 362–371.

Thomas Graf. 2022a. Diving deeper into subregular syntax. *Theoretical Linguistics*, 48:245–278.

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

Thomas Graf. 2022c. Typological implications of tier-based strictly local movement. In *Proceedings of the Society for Computation in Linguistics (SCiL) 2022*, pages 184–193.

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 Aniello De Santo. 2019. Sensing tree automata as a model of syntactic dependencies. In *Proceedings of the 16th Meeting on the Mathematics of Language*, pages 12–26, Toronto, Canada. Association for Computational Linguistics.

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 (SCiL) 2021*, pages 275–285.

Thomas Graf and Connor Mayer. 2018. Sanskrit n-retroflexion is input-output tier-based strictly local. In *Proceedings of SIGMORPHON 2018*, pages 151–160.

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, 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.

Shiori Ikawa, Akane Ohtaka, and Adam Jardine. 2020. Quantifier-free tree transductions. In *Proceedings of the Society for Computation in Linguistics (SCiL)*, volume 3, pages 455–458.

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

Jing Ji and Jeffrey Heinz. 2020. Input strictly local tree transducers. In *Language and Automata Theory and Applications: 14th International Conference, LATA 2020, Milan, Italy*, volume 12038 of *LNCS*, pages 369–381.

Gregory M. Kobele, Christian Retoré, and Sylvain Salvati. 2007. An automata-theoretic approach to Minimalism. In *Model Theoretic Syntax at 10*, pages 71–80.

Marcus Kracht. 2001. Syntax in chains. *Linguistics and Philosophy*, 24:467–529.

Connor Mayer and Travis Major. 2018. A challenge for tier-based strict locality from Uyghur backness harmony. In *Proceedings of Formal Grammar 2018*, pages 62–83, Berlin. Springer.

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

Luigi Rizzi. 1990. *Relativized Minimality*. MIT Press, Cambridge, MA.

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.

John Torr. 2017. Autobank: a semi-automatic annotation tool for developing deep Minimalist grammar treebanks. In *Proceedings of the Demonstrations at the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 81–86.

Erik Zyman. 2023. Raising out of finite clauses (hyperraising). *Annual Review of Linguistics*, 9:29–48.

Definition of ISL mappings with lexical TSL tests

We now allow tree contexts to also contain *tier ports*, which are ports that are indexed with the name of a tier, e.g. $\square_i^T$. We also amend our tree substitution notation to allow for the use of tier ports: $c\{Ti : t\}$ is the result of replacing tier port $\square_i^T$ in context c with t. The indices of tree ports will be interpreted slightly differently from standard ports. Whereas $\square_i$ refers to the (output of the) i-th daughter of the node being rewritten, $\square_i^T$ will refer to the (output of the) node picked out by the i-th TSL test over tier T.

A *lexical TSL test over tier* T is a formula of the form $\phi_T(n, x) := n \lhd_T x \wedge x \in U$, where U is some subset of T. To avoid various complications related to non-determinism, we only consider the special case where $\phi_T(n, x)$ is *deterministic* over some set L of trees. That is to say, for every $t \in L$ and node n of t, there is at most one x such that $\phi_T(n, x)$ is true. We also call $\phi_T(n, x)$ *L-deterministic*. Slightly abusing notation, we let $\phi_T(t, n)$ denote the unique node x (if it exists) such that $\phi_T(n, x)$ holds in t. Finally, we define Φ as a finite family of lexical TSL tests $\phi_{T_1,1}, \ldots, \phi_{T_1,z_1}, \ldots, \phi_{T_k,1}, \ldots, \phi_{T_k,z_k}$ indexed by pairs of tier names and positive natural numbers.

An *ISL rewrite rule with lexical TSL tests over tiers* T_1, …, T_k is a pair $\langle r, \Phi\rangle$ such that $r := \langle i, a, o\rangle$ is an ISL rewrite rule (where o may contain tier ports). We say that $\langle r, \Phi\rangle$ is L-deterministic iff every $\phi_{T,i} \in \Phi$ is L-deterministic. Given such an L-deterministic rule $\rho := \langle r, \Phi\rangle$ and tree $t \in L$, ρ *matches* t at node n with address b iff I) r matches t at address b, and II) for every $\phi_{T,i} \in \Phi$, $\phi_{T,i}(t, n)$ exists. As with ISL rewrite rules, a node at address ba in t can be rewritten by $\rho := \langle\langle i, a, o\rangle, \Phi\rangle$ iff ρ matches t at address b.

A set R of ISL rewrite rules with TSL tests over tiers T_1, …, T_k is *L-deterministic* iff $\{r \mid \langle r, \Phi\rangle \in R\}$ is a deterministic set of ISL rewrite rules and every $r \in R$ is L-deterministic. Note that this excludes any set R containing at least two rules that only differ in their TSL tests.

Given such an L-deterministic set R, $R(t, n)$ denotes the unique output context o for node n in tree $t \in L$. We extend this to t in a recursive fashion: If t contains only node n, then $R(t) := R(t, n)$. If $t := m(d_1, \ldots, d_z)$, then $R(t)$ is

$$\begin{aligned} R(t, m)\{&1 : R(t, d_1), \ldots, z : R(t, d_z),\\ &T_1 1 : R(t, \phi_{T_1,1}(t, m)), \ldots,\\ &T_1 z_1 : R(t, \phi_{T_1,z_1}(t, m)), \ldots,\\ &T_k 1 : R(t, \phi_{T_k,1}(t, m)), \ldots,\\ &T_k z_k : R(t, \phi_{T_k,z_k}(t, m))\} \end{aligned}$$

A tree-to-tree transduction τ with domain D is *deterministic input strictly local with lexical TSL tests* iff there is a finite set R of ISL rewrite rules with TSL tests such that R is deterministic over D and $\tau(t) = R(t)$ for all $t \in D$. In this case, we also call τ an *ISL (tree-to-tree) mapping with lexical TSL tests*.