Volume 26
Issue 1 *Proceedings of the 43rd Annual Penn Linguistics Conference*
Article 16

10-1-2020

Diagnosing Movement via the Absence of C-command Relations

Sabine Laszakovits

Thomas Graf

Diagnosing Movement via the Absence of C-command Relations

Abstract

In this paper, we propose a new diagnostic for movement. It has been argued in the computational linguistics literature that some constraints can be formalized by path constraints on the sequence of their c-commanders (Graf and Shafiei 2019), and that some constraints can be formalized by the tree configuration they appear in (Graf and Heinz 2015). It holds that path constraints are a special case of tree constraints. We make the observation that path constraints cannot account for phenomena that have been argued to involve movement of the element requiring licensing, namely for ATB-extraction and parasitic gaps. However, the reverse does not hold: adjuncts islands and freezing effects, which also involve movement, can be formalized by path constraints. We thus propose the following one-way generalization: Whenever a phenomenon cannot be captured by path constraints, and can be captured by tree constraints, this phenomenon involves movement. We contend that this is not surprising given the fact that constraints on the well-formedness of the Move operation cannot be captured by path constraints, and rather must be captured by tree constraints (Graf 2018).

Diagnosing Movement via the Absence of C-command Relations

Sabine Laszakovits and Thomas Graf[*]

1 Path Constraints and Tree Constraints

Constraints are a core mechanism of many generative formalisms, including Minimalist syntax. Despite their ubiquity, constraints are still poorly understood at a formal level. How should one measure the complexity of constraints? Do the constraints we find in natural languages display uniform complexity? What are the limits of syntactic constraints, and what causes these limitations? A restrictive theory of constraints would make strong typological predictions and provide new diagnostic criteria for syntactic analysis.

These questions have recently been explored in *subregular syntax*. Subregular syntax builds on the notion of subregular complexity, which has been insightfully applied to phonology (see Heinz 2018 and references therein). However, subregular syntax has not yielded a uniform perspective of constraints. Graf and Shafiei (2019) analyze c-command licensing conditions such as Principle A and NPI licensing as constraints on a particular kind of string that lists a node's c-commanders. Graf (2018), on the other hand, applies constraints directly to a compact tree representation called *tree tiers*. These constraints are needed to correctly regulate movement, and they cannot be restated in the string-based terms of Graf and Shafiei (2019). This leaves us with a bifurcation of constraints: *path constraints* apply to strings of c-commanders, whereas *tree constraints* apply to tree-like structures (phrase structure trees, derivation trees, tree tiers, and so on).

This leaves us with the question which class of constraints is more appropriate. In many respects, path constraints are more limited than tree constraints, so on methodological grounds they should be preferred whenever possible. In this paper, we add an empirical dimension to the debate. While the path constraints of Graf and Shafiei (2019) can accommodate a wide range of phenomena, they provably fail in two cases: ATB-extraction, and parasitic gaps. Intuitively, path constraints fail because these two phenomena involve a dependency between elements that occur in independent subtrees and thus do not stand in a c-command relation to each other. Both phenomena are intrinsically tied to movement, though, which is already known from the work of Graf (2018) to be beyond the purview of path constraints. We thus conjecture that phenomena that cannot be captured by path constraints invariably involve movement. If this is correct, then non-definability in terms of path constraints provides indirect evidence for movement.

This paper is organized as follows. In Section 2 we look at examples of path constraints by investigating formalizations of Principle A of Binding Theory in English (Section 2.1), adjunct islands (Section 2.2), and freezing effects (Section 2.3). Whereas the first two build on existing work on the literature, the account of freezing effects is a novel contribution. In Section 3 we show that ATB-extraction (Section 3.1) and parasitic gaps (Section 3.2) cannot be formalized with path constraints, but can be formalized with tree constraints. We proceed to draw a connection between tree constraints and movement in Section 4, which leads us to our generalization that phenomena that cannot be expressed via path constraints invariably revolve around movement. We conclude in Section 5.

2 Phenomena with Path Constraints

As the concept of path constraints is both fairly new and fairly abstract, we cover several examples that illustrate just how much of syntax can be reduced to this simple piece of formal machinery. We start with sketches of how one would formalize anaphora binding (Section 2.1) and adjunct islands (Section 2.2), based on work in Graf and Shafiei (2019) and Shafiei and Graf (2020), respectively. Section 2.3 then presents a novel analysis of freezing effects to illustrate that at least some complex

[*]For helpful comments and discussion, we thank the members of the Math Ling Reading Group at Stony Brook University and the audience and the reviewers of PLC 43.

conditions on movement can also be captured with path constraints.

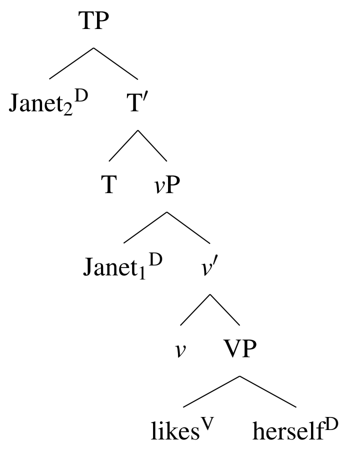
2.1 Anaphor Binding in English

The first phenomenon concerns the licensing of reflexives (anaphor binding, Condition A) in English. According to the standard analysis (Chomsky 1981), reflexives need to be c-commanded by their antecedent within a certain locality domain. In (1), this is fulfilled: the reflexive *herself* is licensed by the c-commanding local ϕ-matching antecedent *Janet*. This contrasts with (2), where a domain boundary intervenes between *herself* and *Janet*. It also differs from (3), where the antecedents *Bill* and *Janet* are both local but *Bill* does not ϕ-match the reflexive and *Janet* does not c-command the reflexive.

(1) Janet likes herself.
(2) * Janet thinks that Bill likes herself.
(3) * Bill invited Janet before liking herself.

Chomsky's (1981) proposal can directly be translated into a path constraint in the sense of Graf and Shafiei (2019). For each node in the tree, one first computes a string that consists of the node followed by all its c-commanders, ordered from lowest to highest. Strictly speaking, Graf and Shafiei (2019) use a slightly different command relation that is sometimes called *d-command* in the computational literature. D-command differs from c-command in that I) a head commands both its specifier and its object, and II) a specifier does not command its head, and III) only heads can stand in d-command relations, not phrases (so if XP c-commands YP, the head X d-commands the head Y). The minor differences between c-command and d-command are immaterial for this paper, but to avoid confusion we will use the term *d-command* whenever we refer to the formal command relation, whereas c-command may still be used as a descriptive term. In (4) we show the paths for all lexical items in the depicted tree ($ explicitly marks the start and end of a string, which is done for mathematical reasons).

(4)


TP
$\text{Janet}_2^{\text{D}}$
T′
T
*v*P
$\text{Janet}_1^{\text{D}}$
v′
v
VP
likes^{V}
$\text{herself}^{\text{D}}$

a. d-commanders(*Janet*$_2$) =
$ *Janet*$_2$ T $
b. d-commanders(T) =
$ T $
c. d-commanders(*Janet*$_1$) =
$ *Janet*$_1$ *v* *Janet*$_2$ T $
d. d-commanders(*v*) =
$ *v* *Janet*$_2$ T $
e. d-commanders(*likes*) =
$ *likes* *Janet*$_1$ *v* *Janet*$_2$ T $
f. d-commanders(*herself*) =
$ *herself* *likes* *Janet*$_1$ *v* *Janet*$_2$ T $

Each lexical item in the tree is now associated with a unique *command string*, or simply *c-string*. We can then proceed to put constraints on the shape of c-strings. Principle A can be expressed as follows:

(5) If a string *s* starts with a reflexive *r*, all of the following must hold:
1. *s* contains an antecedent *a*, i.e. a DP with matching ϕ-features for *r*,
2. *s* must not contain a domain boundary T between *r* and *a*.

For instance, *herself* in (4) would be our reflexive *r*. It is licensed because in the c-string associated with *herself* (4f), *Janet* can take on the role of the antecedent *a*, and no T-head occurs between the two.

Graf and Shafiei (2019) combine this simple system with notions of subregular complexity to more accurately measure the complexity of the path constraints that apply to c-strings. They require that all path constraints must fit into the class IOTSL, which is short for *input output tier-based strictly local* (Graf and Mayer 2018). Intuitively, this means that we may project important elements in a c-string onto a separate tier. On this tier, the constraint must be expressible in terms of a finite list of forbidden substrings. If the tier contains at least one of the forbidden substrings, the c-string that the tier was projected from is deemed illicit.

Principle A would be captured in this system by projecting the delimitation markers \$, the initial reflexive, ϕ-matching antecedents, and T-heads. The resulting tier must not contain the substrings \$*r*\$ or \$*r*T, where *r* is a reflexive. Such illicit tiers can only arise from strings where the TP contains no suitable antecedent for *r*. If, on the other hand, a c-string satisfies Principle A, then the tier must begin with \$*ra*, with *a* some suitable antecedent for *r*. No tier that begin with \$*ra* contains the forbidden substrings \$*r*\$ or \$*r*T, which means that the tier is well-formed and, by extension, so is the c-string. This shows that the IOTSL account does indeed capture Principle A over c-strings.

We include a few additional examples to further illustrate this point. The sentence in (1), whose anaphor has the c-string in (4f) and the tier in (6), is allowed.

(1) *Janet likes herself.*
(4f) d-commanders(*herself*) = \$ *herself likes* $Janet_1$ *v* $Janet_2$ T \$
(6) tier(d-commanders(*herself*)) = \$ *herself* $Janet_1$ T \$ allowed

At the same time, the example in (2) yields a tier with a forbidden substring due to the intervening boundary.

(2) * *Janet thinks that Bill likes herself.*
(7) tier(d-commanders(*herself*)) = \$ *herself* T $Janet_1$ T \$ illicit

Similarly, the example in (3) results in a tier with a forbidden substring due to the lack of a suitable antecedent.

(3) * *Bill* [*invited Janet*] [*before liking herself*].
(8) tier(d-commanders(*herself*)) = \$ *herself* T \$ illicit

Graf and Shafiei (2019) use a similar strategy to account for NPI licensing, and they contend that any attested c-command constraint can be expressed with IOTSL over c-strings. However, this is not quite correct as some constraints require a distinction between containing and non-containing c-commanders. This is illustrated next with the Adjunct Island constraint.

2.2 Adjunct Islands

Adjunct islands (Ross 1967) ban movement out of adjuncts such as *because*-clauses:

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

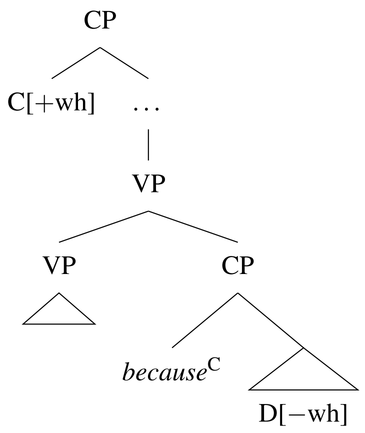
Adjunct islands differ from anaphor licensing in that movement isn't just a complicating factor but the very essence of the phenomenon. Movement-free counterparts of the above sentence (Bobaljik and Wurmbrand 2015) are grammatical:

(10) You fell asleep [because John was reading WHAT]??
(11) You fell asleep [because John was reading Adalbert Stifter].

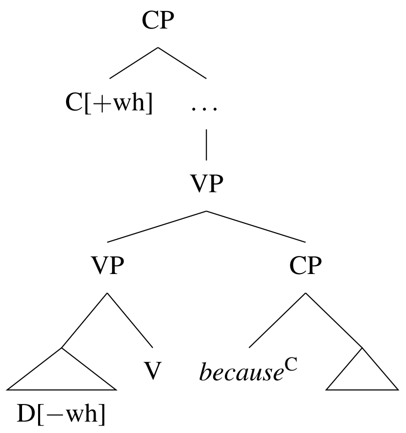
In direct follow-up work to Graf and Shafiei (2019), Shafiei and Graf (2020) show that island constraints can also be captured via path constraints. To do so, it is necessary to extend the contents of the path elements so that they do not only express d-command relations, but also containment. This is because adjunct islands only restrict movers that they contain, not movers that they c-command.

An adjunct island configuration is shown in (12). The feature [−wh] represents the mover and [+wh] the movement-trigger.

(12)



(13)



Looking only at d-command relations, and treating adjuncts like specifiers, the c-string for D[−wh] in (12) would look something like this:

(14) d-commanders(D[−wh]) = \$ D[−wh] ... *because* V ... C[+wh] \$

But this could also be the c-string of D[−wh] in (13), where the mover is not a constituent of the adjunct island. Without containment relations, c-strings cannot correctly distinguish between these two vastly different configurations.

C-strings are thus amended so that they may also include the designated symbol ↑ for containment. Thus, when a head X selects a complement YP, a specifier ZP, and appears with an adjunct UP, the c-string of Y is no longer 'Y Z U X' but rather 'Y Z U ↑ X' to signal that X is the head of the phrase that contains UP, ZP, and YP. With this additional piece of information, adjunct islands are easily expressed in terms of IOTSL. One projects D[−wh], the head X[+wh] with the matching movement feature, and *because* only when is immediately preceded by ↑. This last condition captures the fact that the presence of *because* matters only when it heads a phrase that contains D[−wh]. The resulting tier must have D[−wh] and X[+wh] adjacent to each other. If *because* intervenes between the two, D[−wh] has illicitly moved out of an adjunct island. This account isn't limited to *because*-adjuncts and can be easily generalized to other constructions. One can also allow for specific exceptions such as infinitival *while*-adjuncts as in the example below (Truswell 2007):

(15) Which car did John drive Mary crazy while trying to fix?

We see, then, that even though the original notion of c-strings is too limited to handle all syntactic constraints, the addition of containment extends their empirical coverage by quite a bit. Some constraints even combine c-command and containment in intricate ways. For example, the *that*-trace effect is primarily a containment-based constraint: if X is a subject, it may not move out of a containing CP headed by *that*. But violations of the *that*-trace constraint are ameliorated if an adjunct intervenes.

(16) a. * Who did you say that left?
b. ? Who did you say that without a doubt left?

Intervention here means that the complementizer *that* c-commands an adjunct that c-commands the mover. Both c-command and containment must be encoded in c-strings to account for this behavior.

In sum, c-strings provide a string-based representation of the essential relations that seem to matter for syntactic constraints: c-command and containment. They can not only account for binding effects and NPI licensing, but also for various constraints on movement such as adjunct islands. Next we will show that even complex phenomena such as freezing effects can be captured via c-strings.

2.3 Freezing

Movement is also banned out of moved constituents, which are "frozen" (Culicover and Wexler 1977, Wexler and Culicover 1980). This is illustrated in (17), assuming that English subjects move from Spec,*v*P to Spec,TP.

(17) *? I wonder [who$_i$ [friends of t_i]$_j$ t_j hired Mary].

Using our [−f] [+f] notation from the previous section, we can characterize freezing effects as a ban against nested movement paths. Suppose that in the example above, subject movement is triggered by some feature [±nom], and wh-movement is controlled by [±wh] as before. The c-string of (the lower copy of) *who* in (17) is as follows:

(18) d-commanders(who) =
$ who[−wh] $\uparrow$ of $\uparrow$ friends[−nom] $\uparrow$ D $\uparrow$ *v* $\uparrow$ T[+nom] $\uparrow$ C[+wh] $\uparrow$ wonder $\uparrow$ I[−nom] $\uparrow$ *v* $\uparrow$ T[+nom] $\uparrow$ C $

The span between [−wh] and [+wh] contains a matching pair of [−nom] and [+nom], and crucially [−nom] belongs to a head whose phrase contains the wh-mover [−wh]. This means that wh-movement occurs after the containing phrase undergoes movement to check [−nom] against [+nom], which is exactly the kind of configuration that is forbidden by freezing effects. Over c-strings, then, freezing effects amount to a ban against configurations of the form indicated below.

(19) d-commanders(X[−f]) =
$ X[−f] ... $\uparrow$ Y[−g] ... $\uparrow$ Z[+g] ... $\uparrow$ U[+f] $

This, too, can be expressed in terms of IOTSL, but we omit this step here to focus on the essential insights instead.

While the discussion so far had to gloss over various details, the reader will hopefully be able to appreciate that c-strings can capture a wide range of syntactic phenomena provided they encode both c-command and containment. Since all those constraints over c-strings can be stated in terms of IOTSL, it is tempting to make IOTSL-definability over c-strings an upper bound on the complexity of syntactic constraints. This has various computational implications, but from a linguistic perspective it is most noteworthy for making strong predictions about what a potential syntactic constraint may look like. For example, we do not expect to find an alternating *that*-trace effect where every other *that* in the tree is incapable of inducing this effect. A constraint of this form is beyond the power of IOTSL, and quite generally it cannot be stated over a single c-string because c-strings simply do not provide enough information about the whole tree. The choice of c-strings as the basic representation over which constraints are stated, limits us to path constraints. And path constraints are very restricted in what aspects of a tree they may make reference to.

But as we will discuss next, c-strings (and by extension, path constraints) are unfortunately too limited. There are empirically robust phenomena that provably cannot be accounted for with constraints on c-strings.

3 Phenomena with Tree Constraints

Due to their reliance on c-strings, path constraints cannot capture dependencies that involve elements that reside in distinct subtrees and are not related via containment or c-command. This situation arises in ATB-extraction (Section 3.1) and parasitic gaps (Section 3.2). While path constraints are insufficient for those phenomena, they can be accounted for via tree constraints.

3.1 ATB-extraction

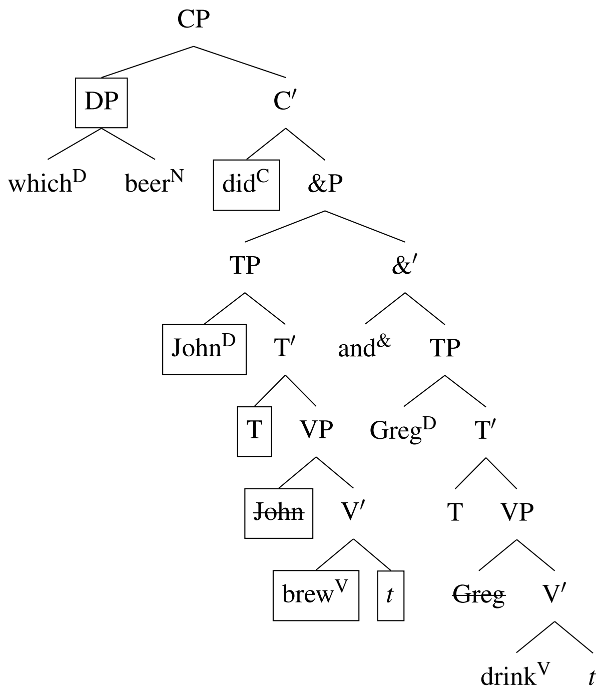
Across-the-board extraction (ATB; Williams 1978) cannot be captured by path constraints, but it can be expressed via tree constraints. ATB-extraction is an exception to the Coordinate Structure Constraint (Ross 1967) such that extraction out of conjuncts is allowed if the extracted element is extracted out of *all* conjuncts. This is illustrated in (20).

(20) a. Which beer did [John brew *t*] and [Greg drink *t*]?
b. * Which beer did [John brew *t*] and [Greg drink wine]?
c. * Which beer did [John brew tea] and [Greg drink *t*]?

Thus, in order to determine the grammaticality of the entire sentence, the grammar needs a constraint that movement out of a coordination is allowed iff the same constituent moves out of each conjunct. This requires a constraint that is aware of the traces in all conjuncts, all at the same time. Considering each trace in isolation is insufficient as the grammaticality of movement hinges on presence/absence of corresponding traces in the other conjuncts. As we show next, this means that the ATB exception to the Coordinate Structure Constraint cannot be stated over c-strings.

A dependency between the two traces in (20a) cannot be established by making reference to the c-commanders of each trace. The first trace's c-commanders are illustrated in the tree in (21) by boxes:

(21)



Note that the c-string of the trace/copy *c* in the first conjunct does not change if we alter the contents of the second conjunct. In particular, whether this conjunct contains a trace is not encoded in the c-string of *c*. Consequently, no path constraint can ever distinguish ATB-movement from non-ATB movement, even if we do not limit path constraints to IOTSL. The failure of path constraints with ATB-extraction is a direct consequence of the restriction to c-strings.

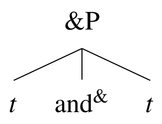
It isn't too surprising, then, that ATB-extraction *can* be accounted for if we abandon c-strings and move from path constraints to tree constraints. With access to the full tree structure, this is fairly trivial. But within the program of subregular syntax, more limited classes of tree constraints have been proposed. As with IOTSL, they are based on constructing tiers which are then further regulated by forbidding specific local configurations. The only difference is that tree constraints build tree tiers instead of string tiers.

We briefly illustrate here what the account in terms of tree tiers looks like. From the tree in (21), we keep only the following nodes:

(22) a. The root &P of the conjunction,
b. every coordination head *and*$^{\&}$, and
c. every mover *t* that is dominated by the root &P and whose final landing site has not yet been reached.

For the present example, this yields the following treelet:

(23)



Then the following must be valid for this treelet, before optional *and*-deletion occurs:

(24) The sequence of daughters of &P follows one of the two patterns below, where x^{+} means "1 or more instances of x in a row":

a. *and*$^{+}$ (There are no movers.)

b. t (*and* t)$^{+}$ (There is exactly one mover in each conjunct.)

This eliminates illicit constructions such as (20b) and (20c) above, where applying the procedure in (22) and (24) would yield the illicit sequences '*t and*', and '*and t*', respectively. The rules in (24) also cover cases with more than two conjuncts.

We conclude, then, that path constraints provably fail to correctly handle ATB-extraction whereas it can be easily done with tree constraints. This holds even if we consider only restricted kinds of tree constraints that are based on tier-projection.

3.2 Parasitic Gaps

Besides ATB-extraction, path constraints are also insufficient for parasitic gaps (Culicover and Postal 2001, Engdahl 1983, Ross 1967, a.m.o.). Parasitic gaps are usually described as the licensing of a gap inside an island, under the condition that the island occurs along a movement path. Alternatively, parasitic gaps may be viewed as a special case of ATB movement: movement out of an island is permitted, provided that the same mover also moves out of a non-island.

This is illustrated in (25). Example (25a) establishes that movement out of a *before*-clause is not possible in general. Then (25b) shows a parasitic gap construction where the constituent moving out of the *before*-clause also moves out of a non-island in the matrix clause. This is well-formed. Finally, (25c) highlights two additional factors: I) mere coindexation with an element in the matrix clause is insufficient to license the parasitic gap, and II) not every instance of movement can license a parasitic gap.

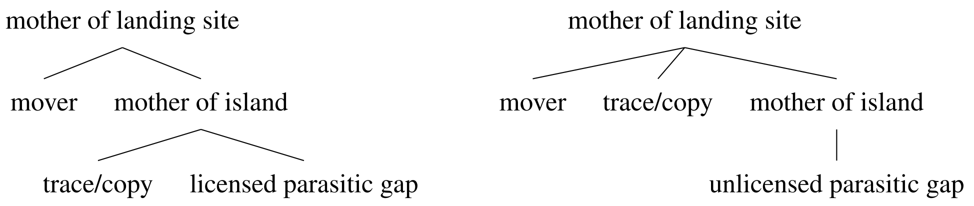
(25) a. * What$_1$ did John [eat a sandwich] [before Mary read t_1]?

b. What$_1$ did John [review a copy of t_1] [before Mary read t_1]?

c. * Who [reviewed a copy of this book$_1$] [before Mary read t_1]?

The ATB-style analysis of parasitic gaps already provides a good clue as to why this phenomenon is beyond the purview of path constraints. The well-formedness of a parasitic gap depends on movement in the matrix clause, but only part of this movement is visible in the parasitic gap's c-string. In particular, this c-string may contain a head with some feature [+f] to indicate that some f-mover will land in this position. But as is well known, this f-mover is not a valid licenser if it originates from a position that c-commands the parasitic gap (Engdahl 1983). In addition, the island containing the parasitic gap must c-command the f-mover. This has two implications. First, it precludes the f-mover from being present in the c-string of the parasitic gap. Second, an f-mover may license the parasitic gap only if the head of the island containing the gap appears in the c-string of the f-mover. So checking the licensing of a parasitic gap at the very least requires inspection of multiple c-strings: that of the gap, and that of every mover that targets a position c-commanding the gap. Path constraints operate on individual c-strings, which means that they cannot inspect multiple c-strings at the same time or relate them to each other. For this reason, the licensing of parasitic gaps cannot be expressed with path constraints.

Just like with ATB, though, tree constraints do provide a solution. Once again it might even be sufficient to consider only tree tiers of a particular kind. On this tier, we keep only the following items: movers, their traces/copies, mothers of landing sites, parasitic gaps, and the mother of the root of each island. Projecting the mother of an island is crucial here to encode in the tier whether a mover is moving from a position c-commanded by the island to a position c-commanding it. This is illustrated below, where the left tree shows a tree tier with a configuration where the parasitic gaps is licensed, whereas the right one is a minimally different variant with an unlicensed gap (the mover and its trace/copy are siblings in the right tier because c-command reduces to sibling hood if all intervening tree material is removed).

(26)


Since the empirical landscape of parasitic gap constructions is quite complex, we refrain from a more precise characterization here. The important issue is that tree constraints definitely furnish sufficient power for parasitic gaps, and further research may reveal that they even fit into the tier-based types of constraints currently favored in subregular syntax. Path constraints, on the other hand, are demonstrably too weak for parasitic gaps.

4 A New Diagnostic for Movement

In the previous two sections we have seen several examples of phenomena and their (non-)formalizability as path constraints. In addition, we may also classify them with respect to whether they necessarily involve movement. This cross-classification leads to a 3/4-signature, illustrated in Table 1.

		Expressible as path constraint?	
		YES	NO
Movement?	NO	Anaphor licensing NPI licensing	Not Attested
Movement?	YES	Adjunct islands Freezing	ATB-extraction Parasitic gaps

Table 1: Empirical basis for the generalization.

Rather than treating the 3/4-signature as an accidental gap, we conjecture that it reveals a fundamental property of syntactic constraints:

(27) If a constraint cannot be captured by path constraints, it involves movement.

The connection between tree constraints and movement is independently motivated from the fact that even the basic principles of the Move operation cannot be expressed by path constraints and require tree constraints instead (see Graf 2018). This furnishes a particular interpretation of the 3/4-signature. ATB-extraction and parasitic gaps involve movement. Tree constraints are necessary to capture Move because path constraints are insufficient. It is then not surprising that path constrains fail ATB-extraction and parasitic gaps, which revolve an aspect of syntax that is already known to be too complex for path constraints.

The reader may object, though, that there are other constraints that involve movement yet can be expressed via path constraints. Adjunct islands and freezing effects, both of which were discussed in this paper, are a clear instance of this. Moreover, Principle A, NPI-licensing effects and other constraints often interact with movement in peculiar ways. But crucially our conjecture only applies in one direction: non-definability with path constraints entails movement, but not necessarily the other way round. Of course this just begs the question as to why the implication isn't a biconditional. There are two potential answers to this.

First, every path constraint is also a tree constraint that only pays attention to a very limited amount of tree structure (this tree constraint might not be expressible via tree tiers, but it would apply over trees). So adjunct islands and freezing effects may simply be instances of tree constraints that are so simple that they can also be stated as path constraints.

Alternatively, there may be a fundamental difference between island constraints and freezing effects on the one hand, and ATB-extraction and parasitic gaps on the other. For instance, the former are about banning movement, whereas the latter are principled exceptions to general bans against movement and/or licensing. Perhaps, then, the relevant distinction isn't so much movement but rather whether one is enforcing constraints or defining exceptions to constraints.

We leave these issues to future research as they do not affect our descriptive generalization in (27). If this generalization should turn out to be correct, it constitutes a novel, computationally grounded heuristic for movement.

5 Conclusions

We have shown that path constraints (Graf and Shafiei 2019) cannot be used to capture all syntactic constraints. Rather, they capture constraints where the licensor and the licensee stand in a c-command or containment relation, but not constraints where they appear in independent subtrees. While this observation on its own is fairly trivial, it becomes a lot more interesting once one takes the specific empirical landscape into account. The only examples of well-established phenomena that are not expressible with path constraints are ATB-extraction out of coordination and parasitic gaps. Looking at a total of six phenomena, we observe that an empirical generalization emerges: All those phenomena that *cannot* be formulated with path constraints involve movement. One may expect, then, that every constraint that is not a path constraint revolves around movement. While basing a generalization on a sample of $n = 2$ exceptions may not have the most footing, we contend that there is independent motivation from the fact that the Move operation itself is not definable as path constraint either. That non-definability with path constraints should then be a heuristic for movement is not surprising.

References

Bobaljik, Jonathan, and Susi Wurmbrand. 2015. Questions With Declarative Syntax Tell Us What About Selection? In *50 Years Later: Reflections on Chomsky's Aspects*, ed. Ángel Gallego and Dennis Ott, volume 77 of *MIT Working Papers in Linguistics*. MIT Press.

Chomsky, Noam. 1981. *Lectures on Government and Binding: The Pisa Lectures*. Studies in Generative Grammar. Foris Publications.

Culicover, Peter, and Paul Postal, ed. 2001. *Parasitic gaps*. MIT Press.

Culicover, Peter, and Kenneth Wexler. 1977. Some syntactic implications of a theory of language learnability. In *Formal Syntax*, ed. Peter Culicover, Thomas Wasow, and Adrian Akmajian, 7–60. New York: Academic Press.

Engdahl, Elisabet. 1983. Parasitic gaps. *Linguistics and Philosophy* 6:5–34.

Graf, Thomas. 2018. Why movement comes for free once you have adjunction. In *Proceedings of CLS 53*, ed. Daniel Edmiston, Marina Ermolaeva, Emre Hakgüder, Jackie Lai, Kathryn Montemurro, Brandon Rhodes, Amara Sankhagowit, and Miachel Tabatowski, 117–136.

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

Graf, Thomas, and Nazila Shafiei. 2019. C-command dependencies as TSL string constraints. In *Proceedings of the Society for Computation in Linguistics*, volume 2, 205–215.

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

Ross, Haj. 1967. Constraints on Variables in Syntax. Doctoral dissertation, MIT.

Shafiei, Nazila, and Thomas Graf. 2020. The Subregular Complexity of Syntactic Islands. In *Proceedings of the Society for Computation in Linguistics (SCiL) 2020*. To appear.

Truswell, Robert. 2007. Extraction from adjuncts and the structure of events. *Lingua* 117:1355–1377.

Wexler, Ken, and Peter Culicover. 1980. *Formal Principles of Language Acquisition*. MIT Press.
Williams, Edwin. 1978. Across-the-Board rule application. *Linguistic Inquiry* 9:31–43.

Sabine Laszakovits
Department of Linguistics
University of Connecticut
Storrs, CT 06269-1145
sabine.laszakovits@uconn.edu

Thomas Graf
Department of Linguistics
Stony Brook University
Stony Brook, NY 11794-4376
mail@thomasgraf.net