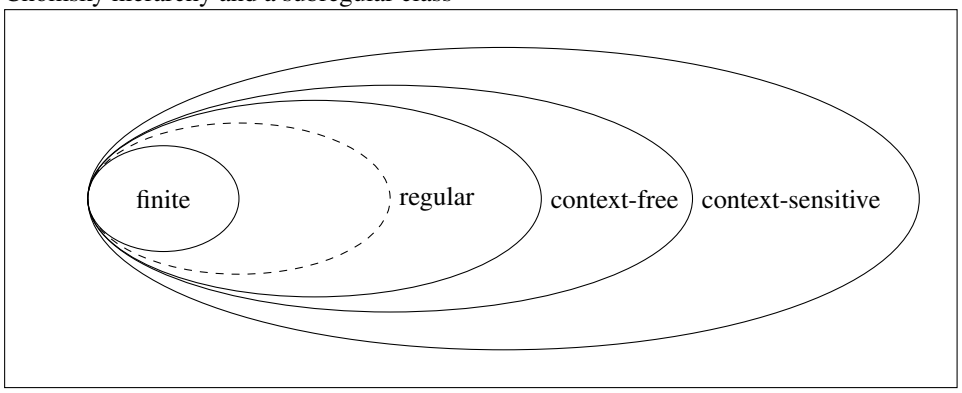
The computational similarity of binding and long-distance consonant dissimilation

Shiori Ikawa and Adam Jardine*

1 Introduction

Chomsky (1956) proposed a hierarchy of grammars in terms of their computational complexity. While Chomsky (1956) originally claimed that natural language is at least context-free, it has been shown that phonology actually falls into smaller classes than the regular class (i.e. subregular classes) as sown in (1), both in terms of their well-formed conditional patterns (Heinz 2018) and transformational patterns (Heinz and Lai 2013, Luo 2017, Chandlee and Heinz 2018). That is, the constraint on the computational complexity of grammar has been shown to be much stronger than context-freeness, at least for phonology.

(1) Chomsky hierarchy and a subregular class



Recently, it has been suggested that not only phonological patterns but also syntactic patterns fall into a class smaller than the regular class, once the structural information is taken into consideration (Graf 2012, Graf and Shafiei 2019, Vu et al. 2019). Graf and Shafiei (2019), for example, showed that binding, when considered as well-formed conditions, fall into a subregular class that is shown to matter for phonological well-formed conditions. The next question to ask is: what about morpho-syntactic transformations, rather than morpho-syntactic well-formedness conditions?

In this paper, we answer this question by examining binding conditions through transformational point of view. Specifically, we take the view that the form of a bound element is morpho-syntactically decided via syntactic conditioning and morphological late insertion (Safir 2014) and show that this process, when logically represented, closely resembles the long-distance dissimilation in the phonological domain. Their resemblance confirms that this morpho-syntactic process falls into *subequential class*, the same subregular class as the one crucial for phonological transformations, when viewed with the correct representation.

This paper is structured as follows. Section 2 introduces the view of binding as a transformation. Section 3 gives background on computational theories of phonology and how they can be extended to syntax, showing the main result of the paper. Section 4 concludes and discusses possible future developments.

2 Binding as morpho-syntactic transformations

Safir (2014), from the point of view of generative syntactic theory, proposes that bound pronouns

*The authors would like to thank the participants of Jardine's spring 2019 seminar on logical characterizations of long-distance patterns at Rutgers and the members of the Rutgers Math Ling reading group for their contributions to this paper.

and reflexive anaphors are underlyingly the same thing, which he calls *D-bound*. D-bound surfaces as a reflexive anaphor when bound within a local domain, but it surfaces as a pronoun when bound non-locally. The binding here is determined by coreference and c-command, and the phase heads (*C* and *v*) define the locality domain. For example, (2) shows the derivation of a sentence that contains a reflexive anaphor, *John likes himself*. The object of the sentence in (2) originates as D-bound+its ϕ-features, as shown in (2a). The step in (2b) show that this D-bound acquires *self* morphology, when it is bound by *John* within the same vP phase. Under the assumption that the morphological form is inserted after the spell-out (Halle and Marantz 1993, 1994, Harley and Noyer 1999:a.o.), the D-bound in (2)is pronounced as *himself*. The binder *John* subsequently moves to Spec TP as shown in (2d), but this movement does not affect the form of the D-bound.

(2) John$_i$ likes *himself*$_i$.
   a. [$_{VP}$ like **Db+3sgM**]
   b. [$_{vP}$ John [$_V$ [$_{VP}$ like **Db+3sgM**]]]
   c. [$_{vP}$ John [$_V$ [$_{VP}$ like **pronoun+3sgM-self**]]]
   d. [$_{vP}$ John [$_V$ [$_{VP}$ like **himself**]]]
   e. [$_{TP}$ John [T [$_{vP}$ ~~John~~ [$_V$ [$_{VP}$ like **himself**]]]]]

On the other hand, (3) shows the derivation of a sentence that contains a pronoun, *John thinks that Mary likes himself*. Similarly to (2), the object of the embedded clause in (3) originates as a D-bound. As there is no co-referring c-commander inside the same phase, and the phase closes before this D-bound gets bound, this D-bound does not acquire the *self*-morphology and is pronounced as a non-reflexive pronoun *him*.

(3) John$_i$ thinks that Mary likes him$_i$.
   a. [$_{VP}$ like **Db+3sgM**]
   b. [$_{vP}$ Mary [$_V$ [$_{VP}$ like **Db+3sgM**]]]
   c. [think [$_{CP}$ [that [$_{TP}$ Mary [$_T$ T [$_{vP}$ ~~Mary~~ [$_V$ [$_{VP}$ like **Db+3sgM**]]]]]]]
   d. [$_{vP}$ John [$_V$ [think [$_{CP}$ [that [$_{TP}$ Mary [$_T$ T [$_{vP}$ ~~Mary~~ [$_V$ [$_{VP}$ like **him**]]]]]]]]]
   e. [$_{TP}$ John [T [$_{vP}$ ~~John~~ [$_V$ [think [$_{CP}$ [that [$_{TP}$ Mary [$_T$ T [$_{vP}$ ~~Mary~~ [$_V$ [$_{VP}$ like **him**]]]]]]]]]]

Thus, borrowing the idea from Safir (2014), the binding data can be captured as morpho-syntactic transformations from D-bound to either pronouns or reflexive anaphors.[1] We can say that D-bounds are transformed into pronouns or reflexive anaphors, conditioned by the positions of their antecedents. Specifically, the D-bounds are transformed into a reflexive anaphor when a co-referent DP c-commands the D-bound in the same local domain, and into a pronoun when a co-referent DP is outside of the local domain. Such transformations can be most naively represented as (4).

(4) a. John$_i$ likes D-bound$_i$ $\mapsto$ John$_i$ likes himself$_i$.
   b. John$_i$ says Mary likes D-bound$_i$ $\mapsto$ John$_i$ says Mary likes him$_i$.

However, the simple representation in (4) fails to include the structural information, which is crucial to represent the c-command by the antecedent. Graf and Shafiei (2019) propose a string-representation of the c-command relationship, as defined in (5). *C-string* (*cs*) of an element *n* is defined over a dependency tree, a tree where each selectee is a daughter of the selector. The resulting string is similar to the bottom-up list of c-commanders of n in the D-structure in the authentic generative syntax.

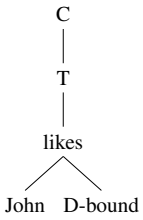
(5) a. If T is a tree s.t. node m has the children $d_1 \cdots, d_i$, d, $d_{i+1} \cdots$, d_n where $n \geq 0$, the immediate command-string $ics(d)$ of d is the string d, $d_{i+1} \cdots$, d_n.
   b. $$cs(n) := \begin{cases} ics(n) & \text{if } n \text{ is the root of } T \\ ics(n) \cdot cs(m) & \text{if } m \text{ is } n\text{'s parent} \end{cases}$$

(Graf & Shafiei, 2019: 208)

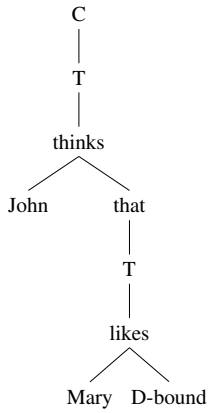
[1] Note that we are making an extensive simplification of Safir's idea here and, to capture the entire binding phenomena, more complex constraints are required. logophor/competition with free pronouns

For the sentences in (2) and (3), the dependency trees á la Graf and Shafiei (2019) look like (6a) and (6b), respectively.[2] Applying the idea of c-string to the current cases, the c-strings of the D-bound in the sentences in (2) and (3) look like (7).

(6) a. *John likes D-bound.*



b. *John thinks that Mary likes D-bound*



(7) a. For the sentence *John likes D-bound*
cs(*D-bound*)= ***D-bound*** *John likes T C*
b. For the sentence *John said Mary likes D-bound*
cs(*D-bound*)= ***D-bound*** *Mary likes T C John say T C*

Now the Binding Conditions A and B are representable as the transformational rules over these c-strings. As only the c-commanders of D-bounds are represented, these strings are capable of capturing the c-command conditions of binding transformations. Given that we are not considering the Minimalist-style derivation, we hereby depart from the notion of phase and will assume, along with Graf and Shafiei (2019) that a finite T is the definer of locality domain for binding. The Binding Conditions A and B can be restated as transformational rules over c-strings : D-bound becomes a reflexive anaphor when a co-referring DP follows it over c-string without any intervening Ts, while D-bound becomes a pronoun when T intervenes between it and the coreferring DPs. The map in (8) shows the binding transformations over the c-strings in (7).

(8) a. ***D-bound*** *John likes T C* $\mapsto$ ***himself*** *John likes T C*
b. ***D-bound*** *Mary likes T C John says T C* $\mapsto$ ***him*** *Mary likes T C John says T C*

Note here that, in binding transformations represented this way, only what follows the D-bound in the c-string matters. For example, one can assume a sentence where a D-bound c-commands another coreferential DP in the same domain. Now the presence or absence of such a DP does not affect the mapping of the c-commanding D-bound to a reflexive anaphor. Thus, what is in the left of D-bound on the c-string does not affect the surface form of D-bound.

Some clarifications are in order here. First, the representation of Condition B here is not equivalent to the standard Condition B effect, which claims that pronouns must be free in the local domain, as first characterized by Chomsky (1981). Among locally unbound pronouns, Safir (2014) distinguishes the bound pronoun and the free pronoun, and considers only the former as instances of D-bounds. What we try to capture in this paper is the transformational patterns of D-bound. Thus, our version of the binding conditions assume that there is always a binder for a D-bound, and the (non-)locality of the binder is the sole factor that determines the form of the D-bound.

Related to the first point, the view of binding as transformations seems to assume the presence of the referential indices in the input, as the form of the D-bound is determined at the end of the transformation based on the specific interpretation of the D-bound. The view of binding conditions as well-formedness conditions (Rogers 1998, Graf and Abner 2012, Graf and Shafiei 2019) could,

[2]These dependency trees abstract away from the features of each node, for the sake of simplification.

on the other hand, do without indices, as a sentence can be judged well-formed as long as there is any indexation that makes the sentence grammatical. To abstract away from the discussions about the validity of assuming indices in the input to the transformation (Rogers 1998, Graf and Abner 2012), we here deal with the cases where there is one and only one ϕ-matching DP in the same sentence as D-bound, and call that DP a binder. We will briefly mention the possibility of index-free representation of the transformation with multiple possible binders in Section 4.

In this section, we have described the binding conditions as transformational rules over c-strings. In the next section, we will delve into the computational complexity of this binding transformation.

3 Computational complexity

The previous section has introduced the view of binding conditions as transformations over c-strings. The question of interest here is whether such binding transformations, which are morpho-syntactic transformations, fall into a subregular class, specifically the same subregular class as is used by the segmental phonological processes. We first introduce in Section 3.1 the subsequential class, a subregular class of function, and how it plays a crucial role in segmental phonology, using long-distance phonological processes as examples. In Section 3.2, we then show that the binding transformations can be logically characterized in a parallel way to long-distance dissimilation process, and the two processes fall into the subsequential class.

3.1 Subsequential class

The *subsequential class* is a subregular class of functions in which transformation of an element can be conditioned by an environment which is unboundedly far from the element but in at most one direction. This class can be formally characterized both by finite state transducers and in terms of the properties of the function itself (Mohri 1997). Crucially for the current purpose, this class has been shown to capture a great deal of segmental phonological processes (Heinz and Lai 2013, Luo 2017, Payne 2017, Chandlee and Heinz 2018). We can then posit a *subsequential hypothesis* that phonological transformations must be subsequential (Heinz and Lai 2013, Heinz 2018, though c.f. Jardine 2016, McCollum et al. 2017).

For example, the long-distance assimilation in Kikonkgo (Ao, 1991) is shown to be a subsequential transformation over string as shown in (9). Here, the /l/ in the suffix becomes [n] when there is a nasal in the root. What is crucial here is that, for the decision of the surface form of this suffix /l/, it can look into its leftward context unboundedly, in the sense that the nasalization of /l/ can be conditioned by a nasal element to its left no matter how many segments intervenes between /l/ and the nasal element. But crucially, what is to the right of the target /l/ does not matter.

(9) Kikongo long-distance nasal assimilation (Ao, 1991)

a.	/ku-toot-ila/	$\mapsto$	[ku-toot-il<u>l</u>a]	'to harvest for'
b.	/ku-du<u>m</u>uk-ila/	$\mapsto$	[ku-du<u>m</u>uk-i<u>n</u>a]	'to jump for'

Long-distance dissimilation as shown in (10) is another example of subsequential transformation (Payne, 2017). Here the /l/ in the suffix becomes [r] when there is another /l/ to its left, no matter how manny segments intervenes between the target /l/ and the conditioning /l/, as shwon in (10a)-(10b). For dissimilation processes, it is further known that, some segments like /r/ or /g/ can behave as blockers. Thus, if there is one of these blockers in between the target /l/ and the conditioning /l/, the change into [r] does not occur, as shown in (10c)-(10d). Note again that what is to the right of /l/ does not condition the change. For example, the change in (11), where an /l/ with another /l/ to its right but not to its left changes into [r], is not uttested.

(10) Latin liquid dissimilation (Czer 2010)

a. /nav-al̲is/ ↦ [nav-al̲is] 'naval'
b. /mi**l**it-al̲is/ ↦ [mi**l**it-ar̲is] 'military'
c. /**l**ito**r**-al̲is/ ↦ [**l**ito**r**-al̲is] 'of the shore'
d. /**l**e**g**-al̲is/ ↦ [**l**e**g**-al̲is] 'legal'

(11) # /nav-al̲is-a**l**/ ↦ [nav-ar̲is-a**l**]

Defining this transformation with a *logical transduction* makes the subsequential nature of this transformation explicitly clear. Logical transductions describe transformational patterns by defining the output properties using the logical descriptions of the input properties (Courcelle 1994). For example, the transformation in (12), where a becomes b when followed by b in the input, can be described by defining the b in the output as the elements which are either b in the input or a followed by b in the input.

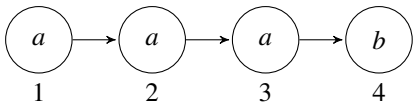
(12) aaaa ↦ aaaa
ab ↦ bb
aaab ↦ aabb
⋮

We use the order-preserving *quantifer-free* (QF) logic over strings (Chandlee and Jardine 2019). This is a predicate logic in which a variable refers to positions in a string and predicates refer to properties of those positions. For example, a predicate $a(x)$ states that position x is an a. In a string *aaab*, $a(x)$ is true when x is interpreted as the first position in the string, but false when it is interpreted as the last position. The *predecessor* and the *successor* functions takes an variable and $p(x)$ and $s(x)$ and returns the elements immediately preceding or following x in the string respectively. The monadic predicates represent properties of elements (e.g., $nasal(x)$ means the element x is a nasal). Using the QF logic, the example transformation in (12) can be described as shown in (13), by defining the property of being an output a (represented as a') and the property of being an output b (represented as b'). (13a) mentions that an element is a in the output if it is a and its successor is not b in the input. (13b) mentions that an element is b in the output if (i) it is b in the input, or (ii) it is a and its successor is b in the input.

(13) a. $a'(x) = a(x) \wedge \neg(b(s(x)))$
b. $b'(x) = b(x) \vee (a(x) \wedge \neg(b(s(x))))$

We will show here how the derivation works, taking the specific case *aaab* ↦ *aabb* in (12) as an example. The input string *aaab* can be represented as shown in (14), where there are positions from 1 to 4, ordered from left to right in this order, and each is labeled with its property (*a*/*b*). The arrow represents the successor relationship among the elements. The truth values for the statements $a(x)$, $b(x)$ and $b(s(x))$ for each of these elements are summarized in the upper half of (15).[3] These truth values and the definition in (13) decides which output properties, a' or b', is assigned to each element as shown in the lower half of (15). As the element 1 and 2 makes $a'(x)$ true and the element 3 and 4 makes $b'(x)$ true, the resulting string is *aabb*.

(14)



(15) Derivation of *aaab* ↦ *aabb*

	a	a	a	b
	1	2	3	4
$a(x)$	T	T	T	F
$b(x)$	F	F	F	T
$b(s(x))$	F	F	T	F
$a'(x)$	T	T	F	F
$b'(x)$	F	F	T	T

Chandlee and Jardine (2019) further augment the QF logic with the recursive predicates (*least fixed point (lfp)* predicates; see Chandlee and Jardine (2019) for the formal definition). While the

[3] We assume here that the last element (4) is succeeded by the word boundary marker, which makes $b(s(x))$ false.

successor function only allows us to refer to the things that are immediately followed by something, with the recursive predicates and with the successor function, we can define a set of things that are (not immediately) followed by something by recursively applying the successor function. For example, in the transformation in (16), all the *a*s followed by *b* become *b*, even the ones not-immediately followed by *b*. The recursive predicate in (17a) is defined as the property of (i) being immediately followed by *b* (the base case), or (ii) being followed by an element satisfying this predicate (the recursive case). This results in a predicate describing the property of being (not necessarily immediately) followed by *b*. (17b) then defines the output *a* as the input *a* that does not satisfy (17a). (17c) defines the output *b* as (i) the input *b* or (ii) the input *a* that satisfies (17a). Thus, the string *aaab* is mapped to *bbbb*, as all the *a*s in this string satisfies *followed-by-b*(x) and gets mapped to *b*, as the derivation shown in (18).

(16) aaaa $\mapsto$ aaaa
ab $\mapsto$ bb
aaab $\mapsto$ bbbb
$\vdots$

(17) a. *followed-by-b*(x)=$b(s(x)) \vee$ *followed-by-b*$(s(x))$
b. $a'(x)$=$a(x) \wedge \neg$*followed-by-b*(x)
c. $b'(x)$=$b(x) \vee (a(x) \wedge$*followed-by-b*$(x))$

(18) Derivation of *aaab* $\mapsto$ *bbbb*

	a	a	a	b
	1	2	3	4
$a(x)$	T	T	T	F
$b(x)$	F	F	F	T
$b(s(x))$	F	F	T	F
followed-by-b(x)	T	T	T	F
$a'(x)$	F	F	F	F
$b'(x)$	T	T	T	T

The QF logic enriched with this predicate is called QFLFP (Chandlee and Jardine 2019). Crucially, Chandlee & Jardine show that the functions expressible with QFLFP logical transductions with either the predecessor function p or the successor function s, but not both, to be the subset of subsequential functions. Intuitively, QFLFP logic can describe the transformation conditioned by an unboundedly far element with the recursive predicate and, with the limitation that p and s cannot be both used at the same time, such unbounded effect is limited to one direction, ensuring that the subsequential nature of QFLFP logical transductions. Thus, if we can describe the long-distance dissimilation using QFLFP logic, it formally proves that the long-distance dissimilation is a subsequential process.

Applying the QFLFP logic, the long-distance dissimilation pattern in (10) can be described as shown in (19). We can define exactly when the /l/ in *-alis* appears as [r] in the output with the following monadic predicates. The formula in (19a) recursively defines exactly when an element x follows an /l/ without intervening /r/: As the base case, this an element satisfies this predicate when either its immediate predecessor is /l/ ($l(p(x))$). Note that, in defining the recursive case, this predicate has to be sensitive to the existence of the intervening /r/ (the blocker). Thus, the recursive case is defined as the cases where the immediate predecessor satisfies *follows-l* AND is being not a blocker (/r/ or non-coronals) ($\neg(r(p(x)) \vee$ *non-coronal*$(p(x))) \wedge follows\text{-}l(p(x))$). The formula in (19b) then defines exactly when a segment surfaces as [r] in the output: either it was an /r/ in the input (r(x)) or it was an /l/ *and* it followed an /l/ without intervening blockers ($l(x) \wedge follows\text{-}l(x)$).[4]

(19) a. $follows\text{-}l(x) := l(p(x)) \vee (\neg(r(p(x)) \vee noncoronalc(p(x))) \wedge follows\text{-}l(p(x)))$
b. $\mathrm{r}'(x) := \mathrm{r}(x) \vee (\mathrm{l}(x) \wedge follows\text{-}l(x))$

[4]This formula describes the alternation in the suffix, and is not intended to describe the general alternation pattern of [l]s and [r]s in Latin.

The mappings in (20) can be correctly captured, as shown in the derivation in (21) and (22). Crucially, although the /l/ in the suffix in (21) makes *follows-l*(x) true, the /l/ in the suffix in (22) does not, as the intervening element, 3, satisfies *noncoronalc*(x) and stops the recursive appication of follows-l(x). Thus, the former is mapped to the output [r], while the latter remains [l].

(20) Latin liquid dissimilation

/mi**l**it-a**l**is/	$\mapsto$	[mi**l**it-a**r**is]	'military'
/**leg**-a**l**is/	$\mapsto$	[**leg**-a**l**is]	'legal'

(21) Derivation of /milit-alis/ $\mapsto$ [milit-aris]

	m	i	l	i	t	a	l	i	s
	1	2	3	4	5	6	7	8	9
$l(x)$	F	F	T	F	F	F	T	F	F
$r(x)$	F	F	F	F	F	F	F	F	F
noncoronalc(x)	T	F	F	F	F	F	F	F	F
follows-l(x)	F	F	F	T	T	T	T	F	F
$l'(x)$	F	F	T	F	F	F	**F**	F	F
$r'(x)$	F	F	F	F	F	F	**T**	F	F

(22) Derivation of /leg-alis/ $\mapsto$ [leg-alis]

	l	e	g	a	l	i	s
	1	2	3	4	5	6	7
$l(x)$	T	F	F	F	T	F	F
$r(x)$	F	F	F	F	F	F	F
noncoronalc(x)	F	F	T	F	F	F	F
follows-l(x)	T	T	T	F	F	F	F
$l'(x)$	T	F	F	F	**T**	F	F
$r'(x)$	F	F	F	F	**F**	F	F

Thus, (19) shows that the long-distance dissimilation pattern in Latin can be described with QFLFP logic using only the predecessor function, but not the successor function. This confirms that long-distance dissimilartion is a subsequential pattern. Now our question is whether the binding as characterized in Section 2 can be described in a parallel way with QFLFP logic?

3.2 Logical description of Binding Transformations

Recall that the transformational rules that we try to describe is the one over the c-string transformation shown again in (23): D-bound becomes a reflexive anaphor when a coreferring DP follows it over c-string without any intervening Ts, while D-bound becomes a pronoun when T intervenes between it and the coreferring DPs.

(23) a. ***D-bound*** *John likes T C* $\mapsto$ ***himself*** *John likes T C*
b. ***D-bound*** *Mary likes T C John says T C* $\mapsto$ ***him*** *Mary likes T C John says T C*

(24) captures the transformation in (23) with a logical transduction. The predicate *binder* refers to the property of being the sole ϕ-matching DP to the D-bound (See the discussion at the end of Section 2). (24a) recursively defines when an element x locally precedes a binder: First, (24a) defines the recursive predicate. The base case is when something immediately precedes the binder. The recursive part is true when the successor satisfies this recursive predicate AND is not the blocker T. This amounts to describe the property of being followed by the binder without an intervening T. Note that this closely parallels the recursive predicate we defined for the long-distance dissimilation in (19a). The formulae in (24b) define what surface as reflexives or pronouns. An element x surfaces as a reflexive when it is a D-bound in the input and locally precedes the binder, and surfaces as a pronoun when it is a D-bound in the input and does not locally precedes the binder. The formula in (24a) is clearly parallel to (19a): x locally precedes the binder either when its immediate successor is the binder ($binder(s(x))$) or its immediate successor is not a T and locally precedes the binder ($\neg T(s(x)) \wedge locally\text{-}prec\text{-}binder(s(x))$). Crucially, as (24) uses only successor and (recursively-defined) QF predicates, it is subsequential.

(24) a. $\mathit{locally\text{-}prec\text{-}binder}(x) := \mathit{binder}(s(x)) \vee (\neg T(s(x)) \wedge \mathit{locally\text{-}prec\text{-}binder}(s(x)))$
b. $\mathit{reflexive}'(x) := \mathit{D\text{-}bound}(x) \wedge \mathit{locally\text{-}precedes\text{-}binder}(x)$
$\mathit{pronoun}'(x) := \mathit{D\text{-}bound}(x) \wedge \neg \mathit{locally\text{-}precedes\text{-}binder}(x)$

The tables show the derivations for the examples in (23). The crucial difference between the two is that, although the element 1, the D-bound, makes *locally-prec-binder*(x) true in (25), the D-bound in (26) does not, given that the intervening T stops the spreading of the property *locally-prec-binder* to the D-bound. Thus, the D-bound satisfies the output predicate *reflexive'* in (25), but not in (26). Conversely, the D-bound satisfies the output predicate *pronoun'* in (26), but not in (25).

(25) Derivation of (23a)

	D-bound	John	likes	T	C
	1	2	3	4	5
D-bound(x)	T	F	F	F	F
binder(x)	F	T	F	F	F
$T(x)$	F	F	F	T	F
locally-prec-binder(x)	T	F	F	F	F
reflexive'(x)	**T**	F	F	F	F
pronoun'(x)	F	F	F	F	F

(26) Derivation of (23b)

	D-bound	Mary	likes	T	C	John	says	T	C
	1	2	3	4	5	6	7	8	9
D-bound(x)	T	F	F	F	F	F	F	F	F
binder(x)	F	F	F	F	F	T	F	F	F
$T(x)$	F	F	F	T	F	F	F	T	F
locally-prec-binder(x)	F	F	F	T	T	F	F	F	F
reflexive'(x)	F	F	F	F	F	F	F	F	F
pronoun'(x)	**T**	F	F	F	F	F	F	F	F

The definition in (24) makes explicit the similarities between the transformation in (8) and consonant dissimilation: Crucially, a D-bound searches for the binder uni-directionally in the sense that it only looks to the right. It is also unbounded in the sense that it does not care how many non-T elements are in between the D-bound and the binder. These properties indicates the binding transformation falls into the subsequential class, similarly to the consonant dissimilation.

4 Conclusions and Discussions

We have shown that long-distance consonant dissimilation and binding conditions can be captured with quantifier-free logic over strings enriched with recursive predicates. Crucially, both used either one of predecessor or successor functions. This ensures that both are in the subsequential class. This result supports the hypothesis that both syntax and phonology fall into the same subregular class even in terms of transformations.

However, this work has some limitations. First in this work, we assumed that there is always one binder and the transducer can identify it. But what happens if there are multiple possible binders? As shown in (27), the form of the D-bound depends on which DP is the antecendent of the D-bound. One way to capture this observation is to introduce the indices in the syntax (see the discussion in Rogers (1998) and Graf and Abner (2012)). Another way, however, is to stick to the index-free syntax and consider that the transformation becomes optional in (27): If there is both a local possible binder and a non-local possible binder for a D-bound as shown in (27), the D-bound can be optionally realized either as a pronoun or a reflexive anaphor.

(27) Mary thinks Ann likes her/herself.

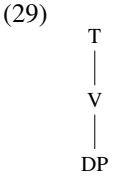
If we take the latter option, the situation undergoes further complication when we consider more than one D-bounds: the maximum number of surface pronouns are affected by the number of non-local possible antecedents (Graf and Abner 2012, Graf and Shafiei 2019)

(28) a. #Mary thinks she likes her.
b. Mary thinks Ann knows she likes her.

It needs further investigation how such optionality should be represented, and whether the representation of the optionality affects the computational class the binding transformation falls into.[5]

Second, in this work, we applied the idea of c-strings (Graf and Shafiei 2019) to represent the structural information in the binding transformations. C-strings are useful in that they allow direct comparisons between syntactic phenomena, which refer to structural information, and the phonological phenomena represented in a string format. On the other hand, the application of the idea to represent transformations raises another issue: the c-string of which element should be transformed, and how does it affect the final representation of the entire sentence? This is especially problematic for the transformations of an element conditioned by its preceding elements over c-strings (i.e. conditioned by the elements c-commanded by it in a tree). For example, consider a hypothetical transformational pattern where V becomes X when it selects a DP. This can be represented as a logical transduction over c-strings which maps V to X when its predecessor is a DP. Taking the derivation tree in (29) as an example, considering $cs(V)$ in (30a), V does not get transformed into X, as no DP precedes it. However, considering $cs(DP)$ in (30b), V gets transformed into X, as DP precedes it there.

(29)



(30) a. *cs*(V)= V T
b. *cs* (DP)= DP V T

The current case is not entirely free from the problem of whose c-strings to be considered either: Suppose we represent the choice of antecedents in (31) as optionality, given the discussion about (27) above. The two c-strings in (32) can choose a different option regarding whether the D-bound is mapped to a reflexive anaphor or a pronoun, resulting in a contradiction with each other in terms of what the D-bound is mapped to.

(31) John said that Bill told *D-bound* that Mary is kind.
(32) a. *cs*(that)= *that D-bound Bill show T C John T C*
b. *cs*(D-bound)=*D-bound Bill show T C John T C*

Thus, the current attempt reveals the limitation of using c-strings in transformations. Several works define the tree transformation classes corresponding to the classes useful for phonological transformations (Graf 2020, Ikawa et al. 2020, Ji and Heinz 2020), and the examination of the binding transformation over trees would be a next step.

References

Bhaskar, Siddharth, Jane Chandlee, Adam Jardine, and Christopher Oakden. 2020. Boolean monadic recursive schemes as a logical characterization of the subsequential functions. In Language and Automata Theory and Applications - LATA 2020, ed. Alberto Leporati, Carlos Martín-Vide, Dana Shapira, and Claudio Zandron, Lecture Notes in Computer Science, 157–169. Springer.

Chandlee, Jane, and Jeffrey Heinz. 2018. Strictly locality and phonological maps. LI 49:23–60.

Chandlee, Jane, and Adam Jardine. 2019. Autosegmental input-strictly local functions. Transactions of the Association for Computational Linguistics 7:157–168.

[5]The conjecture is that, if the optionality is described using the set parameter (Engelfriet and Hoogeboom 2001), the examples in (28) require a slightly broader class of transformations than QFLFP describes, but still fall under the subsequential class, by being logically describable with BMRS (Bhaskar et al. 2020).

Chomsky, Noam. 1956. Three models for the description of language. IRE Transactions on information theory 2:113–124.

Chomsky, Noam. 1981. Lectures on government and binding, foris, dordrecht.

Courcelle, Bruno. 1994. Monadic second-order definable graph transductions: a survey. Theoretical Computer Science 126:53–75.

Engelfriet, Joost, and Hendrik Jan Hoogeboom. 2001. MSO definable string transductions and two-way finite-state transducers. ACM Transations on Computational Logic 2:216–254.

Graf, Thomas. 2012. Locality and the complexity of minimalist derivation tree languages. In Formal grammar, 208–227. Springer.

Graf, Thomas. 2020. Curbing feature coding: strictly local feature assignment. Proceedings of the Society for Computation in Linguistics 3:362–371.

Graf, Thomas, and Natasha Abner. 2012. Is syntactic binding rational? In Proceedings of the 11th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+11), 189–197.

Graf, Thomas, and Nazila Shafiei. 2019. C-command dependencies as tsl string constraints. In Proceedings of the Society for Computation in Linguistics (SCiL) 2019, 205–215.

Halle, Morris, and Alec Marantz. 1993. Distributed morphology and the pieces of inflection. hale, k. & sj keyser (eds.), the view from building 20.

Halle, Morris, and Alec Marantz. 1994. Some key features of distributed morphology. MIT working papers in linguistics 21:88.

Harley, Heidi, and Rolf Noyer. 1999. Distributed morphology. Glot international 4:3–9.

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

Heinz, Jeffrey, and Regine Lai. 2013. Vowel harmony and subsequentiality. In Proceedings of the 13th Meeting on Mathematics of Language, ed. Andras Kornai and Marco Kuhlmann. Sofia, Bulgaria.

Ikawa, Shiori, Akane Ohtaka, and Adam Jardine. 2020. Quantifier-free tree transductions. In Proceedings of the Society for Computation in Linguistics, volume 3.

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

Ji, Jing, and Jeffrey Heinz. 2020. Input strictly local tree transducers. In International Conference on Language and Automata Theory and Applications, 369–381. Springer.

Luo, Huan. 2017. Long-distance consonant agreement and subsequentiality. Glossa 2.

McCollum, Adam, Eric Baković, Anna Mai, and Eric Meinhardt. 2017. Conditional blocking in Tutrugbu requires non-determinism: implications for the subregular hypothesis. In Paper presented at NELS 48.

Mohri, Mehryar. 1997. Finite-state transducers in language and speech processing. Computational Linguistics 23:269–311.

Payne, Amanda. 2017. All dissimilation is computationally subsequential. Language: Phonological Analysis 93:e353–371.

Rogers, James. 1998. A Descriptive Approach to Language-Theoretic Complexity. Chicago: Center for the Study of Language and Information.

Safir, Ken. 2014. One true anaphor. Linguistic Inquiry 45:91–124.

Vu, Mai Ha, Nazila Shafiei, and Thomas Graf. 2019. Case assignment in TSL syntax: A case study. In Proceedings of the Society for Computation in Linguistics (SCiL) 2019, 267–276.

Department of Linguistics
Rutgers University
New Brunswick, NJ 08901
shiori.ikawa@rutgers.edu
adam.jardine@rutgers.edu