Towards a Formal Description of NPI Licensing Patterns*

Mai Ha Vu
University of Delaware
maiha@udel.edu

Abstract

This paper is a formal study of a simplified version of Negative Polarity Item (NPI) licensing requirements in two languages, English and Hungarian. In the framework of Model-Theoretic Syntax, using logical formalisms defined over tree-languages, I show that neither pattern can be described with Tier-based Strictly Local (TSL) constraints only, and suggest that they need a more complex logical formula. In particular, Hungarian patterns can be described using a combination of Tier-based Strictly 2-Local constraints over dominance relations and Locally 1-Testable constraints over the left-of relations between nodes. For English, there are no sufficient local constraints, either with or without tiers. As part of the analysis, I give a definition of a generalized tree-language that uses Tier-based 2-Local constraints over dominance relations, while it remains underspecified for left-of relations.

1 Introduction

Model-theoretic syntax is a way to study linguistic structures formally by describing them in terms of logical constraints, rather than in terms of sequences of derivational steps. While its roots can be traced back to early studies in generative grammar, it gained prominence with James Rogers' 1998 work, *A Descriptive Approach to Language-Theoretic Complexity* (Pullum, 2007).

Rogers' (1998) results showed that a significant portion of Government and Binding Theory can be described with a version of Monadic Second-order (MSO) constraints over phrase-structure trees. Incidentally, structures that can be described with MSO logic are members of the regular class of languages in terms of complexity (Rogers and Pullum, 2011).

It is known, however, that many regular languages are not plausible patterns in human natural language (Heinz and Idsardi, 2013). As an example, the *even-a* language, which is defined as a set of strings that can only contain an even number of *a*s, is widely considered implausible. As a result, recent work has focused on identifying *subregular* regions relevant to natural language. While progress has been made on phonotactic patterns (Heinz, 2009; Heinz, 2010; Jardine, 2016) and phonological transformations (Chandlee, 2014), less has been said in this regard about syntactic patterns.

Relevant work on syntax in this vein has been done by Thomas Graf, who has argued that most linguistic patterns, including syntactic and morphological ones, fit in the Tier-based Strictly Local (TSL) class (Graf and Heinz, 2015; Graf, 2017). While the TSL class was originally used to describe stringsets (Heinz et al., 2011), we use the class in a more abstract way, and apply it to tree-sets in the current paper.

Because trees are two-dimensional structures with two types of ordering relations in them (Rogers, 2003), the type of ordering relation over which a certain class of language applies has to be specified. The TSL tree-language as described by Graf and Heinz (2015) has TSL constraints over

*I thank Jeffrey Heinz, Thomas Graf, Hossep Dolatian, Kristina Strother-Garcia, and the anonymous reviewers for their thoughtful and insightful feedback on earlier drafts of this paper. All errors are my own.

Proceedings of the Society for Computation in Linguistics (SCiL) 2018, pages 154-163.
Salt Lake City, Utah, January 4-7, 2018

both the dominance and left-of ordering relations. This paper provides a more generalized definition of Tier-based Strictly 2-Local tree-grammars: tree-grammars where the constraints are Tier-based Strictly 2-Local over dominance ($\text{TSL}_2^{\triangleright}$), but can be of different complexity over left-of relations.

We then demonstrate that a particular type of pattern where the existence of one item in the structure requires the existence of another one, formalized as $a \rightarrow b$, cannot be described with TSL constraints in the sense of Graf and Heinz (2015). NPIs fall into this category of patterns, as an NPI cannot occur without a licensor.

The scope of this paper is restricted to describing well-formed surface structures, without any assumption of underlying features or syntactic movement. We are thus agnostic about any in-depth theory of NPI-licensing, and are not addressing specific proposals suggesting movement or agreement (cf. Giannakidou and Zeijlstra (2016)); neither do we look at proposed Logical Forms of these sentences, which might differ from the observed surface word orders.

Lastly, the choice of syntactic data structure needs a few words. Two common data structures used to describe syntactic structures are phrase-structure trees and derivation trees. For a detailed discussion of the two, the reader is referred to Stabler (1997). We choose to use phrase-structure trees as the data structure for the sentences discussed in this paper, instead of derivation trees. Graf (2013) gives an in-depth analysis of the nature of syntactic constraints, both over phrase-structure trees and derivation trees. He shows that representational constraints (i.e. those over phrase-structure trees), are subsumed by translocal constraints (i.e. those over derivation trees). We thus believe that modeling NPI-licensing with constraints over phrase-structure trees will not take away from the overall generalizability of our results regarding the complexity of necessary constraints in natural language syntax.

The paper is organized as follows. Section 2 states the definitions of key concepts needed to understand the discussion in the rest of the paper. Section 3 introduces the syntactic data in question: NPI-licensing in English and Hungarian. Section 4 shows that these patterns need TLT and First-Order Logic to be described. Section 5 concludes.

2 Preliminaries

2.1 Strictly Local and Locally Testable Stringsets

These definitions of Strictly Local (SL) and Locally Testable (LT) stringsets are largely based on Heinz et al. (2011), Rogers and Pullum (2011), and Rogers et al. (2013). We assume familiarity with monadic second-order (MSO) logic (Enderton, 2001).

First, k-factors over strings are defined below. Let Σ be the alphabet, and Σ^* be all strings of finite length over Σ. Then string u is a factor of string w iff $(\exists x, y \in \Sigma^*)$ such that $w = xuy$. If $|u| = k$, then u is a k-factor of w. The function F_k maps a string to a set of k-factors within it:

$$F_k(w) = \{u | u \text{ is a } k\text{-factor of } w\}$$

A Strictly k-Local (SL_k) grammar for a string language is understood as a list of possible k-factors in the language, or equivalently, a list of banned k-factors.

Definition 1 *(Strictly Local Stringsets)*

$\mathcal{G}$, a Strictly k-Local description over some alphabet Σ, is a set of k-factors of $\Sigma \cup \{\rtimes, \ltimes\}$, where $\rtimes$ and $\ltimes$ mark the beginning and ending of a string, respectively.

$$\mathcal{G} \subseteq F_k(\rtimes \cdot \Sigma^* \cdot \ltimes)$$

A string w satisfies $\mathcal{G}$, iff the set of k-factors of the augmented string $\rtimes \cdot w \cdot \ltimes$ is a subset of $\mathcal{G}$:

$$w \models \mathcal{G} \Leftrightarrow F_k(\rtimes \cdot w \cdot \ltimes) \subseteq \mathcal{G}$$

The stringset licensed by a description $\mathcal{G}$ is the set of words that satisfy it.

$$L(\mathcal{G}) \stackrel{def}{=} \{w | w \models \mathcal{G}\}$$

A set of strings is Strictly k-Local (SL_k) iff it is $L(\mathcal{G})$ for some strictly k-local definition of $\mathcal{G}$. It is Strictly Local iff it is SL_k for some k.

Next, the definition of local k-expressions is given below.

Definition 2 *(Local k-expressions)*

The language of k-expressions is the smallest set including the following forms, with the intended semantics indicated.

- *Atomic formulae: $f \in F_k(\Sigma^*)$ is a k-expression.*

- *Conjunction: If φ_1 and φ_2 are k-expressions, then $(\varphi_1 \wedge \varphi_2)$ is a k-expression.*
- *Negation: If φ_1 is a k-expression, then $(\neg\varphi_1)$ is a k-expression.*

If w is a string and φ a k-expression, then
$w \models \varphi \overset{def}{\Leftrightarrow}$
$$\begin{cases} \varphi = f \in F_k(\Sigma^*) \text{ and } f \in F_k(w) \\ \varphi = (\varphi_1 \wedge \varphi_2) \text{ and } w \models \varphi_1 \text{ and } w \models \varphi_2 \\ \varphi = (\neg\varphi_1) \text{ and } w \not\models \varphi_1 \end{cases}$$

Now we can define Locally Testable Stringsets with the help of k-expressions.

Definition 3 *(Locally Testable Stringsets)*
A stringset L over Σ is k-Locally Testable (LT_k) iff there is some local k-expression φ over Σ (for some k) such that L is the set of all strings that satisfy φ.

$$L = L(\varphi) \overset{def}{=} \{w \in \Sigma^* | w \models \varphi\}$$

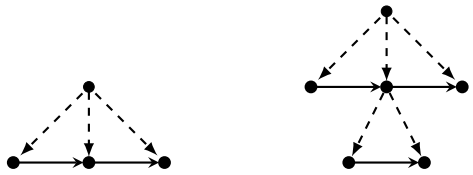
A stringset is LT iff it is LT_k for some k.

Notice that implicational statements can be derived from k-expressions, because $a \rightarrow b$ is equivalent to $\neg(a \wedge \neg b)$.

2.2 Tree languages

Our understanding of trees is based on the idea of *multi*-dimensional trees, as discussed in Rogers (2003). For the purposes of this paper, we exclusively work with 2-dimensional trees, and thus restrict our formal descriptions to them.

The basic intuition is as follows. Strings are one-dimensional trees, whose nodes are related to each other via one-dimensional successor relations. To add a second dimension, we first build a *local* tree (a tree of at most one depth) by connecting a single point *a* to each node in a one-dimensional tree *S* through second-dimensional successor relations (Figure 1a). The adjoined point is called the *root* in this case, and the nodes in *S* are the *yield*. A *composite* tree, where trees have depths greater than one, can be built by identifying the root of one local tree with some point in the yield of another (Figure 1b). In the trees in Figure 1, the solid lines represent the one-dimensional successor relations, and the dashed lines represent the second-dimensional successor relations.



(a) 2-dimensional local tree (b) 2-dimensional composite tree

Figure 1: Two-dimensional trees

We give our formal definition for two-dimensional trees within the model-theoretic framework. Model theory provides a way to describe a particular object using mathematical logic. A model requires a *signature* and a set of *logical statements*. Along with the usual logical connectives, $x \approx y$ denotes that x and y are equivalent.

The signature of the two-dimensional tree-model for linguistic trees is shown in Figure 2. Let $\Sigma = \Sigma_{Cat} \cup \Sigma_{Lex}$ be the alphabet, where Σ_{Cat} is the set of syntactic categories, and Σ_{Lex} is the set of lexical items. We write $\sigma(x)$ if node x is labeled with σ, for all $\sigma \in \Sigma$.

$$\langle \mathcal{D}, \prec, \triangleright, \mathcal{L}_\sigma \rangle_{\sigma \in \Sigma}, \text{ where}$$

- $\mathcal{D}$ is the finite domain
- $\prec$ is a binary ordering relation *immediate left-of*
- $\triangleright$ is a binary ordering relation *immediate dominance*
- $\mathcal{L}_\sigma$ is a set of unary relations for labeling elements in $\mathcal{D}$ with σ for all $\sigma \in \Sigma$

Figure 2: Model for two-dimensional trees

Following Rogers (2003), $\triangleright^*$ is defined as the reflexive transitive closure, and $\triangleright^+$ as the transitive closure of $\triangleright$. This is explicitly monadic second-order definable through $Branch$: a set of nodes that are upwards closed with regard to, and linearly-ordered by the immediate dominance relation, $\triangleright$. The *depth* of any tree then can be understood as the length of the longest Branch in the tree.

(1) $Branch(X) \equiv (\forall x, y)[(X(x) \wedge y \triangleright x) \rightarrow X(y)] \wedge (\forall x, y, z)[(X(x) \wedge X(y) \wedge X(z) \wedge$

$x \rhd y \wedge x \rhd z) \rightarrow y \approx z]$

(2) $x \rhd^* y \equiv (\forall X)[(Branch(X) \wedge X(y)) \rightarrow X(x)]$

(3) $x \rhd^+ y \equiv x \rhd^* y \wedge x \not\approx y$

The predicates $\prec^*$ and $\prec^+$ are definable in a similar fashion.

(4) $String(X) \equiv (\forall x, y)[(X(x) \wedge y \prec x) \rightarrow X(y)] \wedge (\forall x, y, z)[(X(x) \wedge X(y) \wedge X(z) \wedge x \prec y \wedge x \prec z) \rightarrow y \approx z]$

(5) $x \prec^* y \equiv (\forall X)[(String(X) \wedge X(y)) \rightarrow X(x)]$

(6) $x \prec^+ y \equiv x \prec^* y \wedge x \not\approx y$

Based on $\prec^*$, we also can also define the inherited left-of relation, $<^*$. If z is left-of w in a tree, then all nodes that are reflexively dominated by z are inherited left-of all nodes that are reflexively dominated by w.

(7) $x <^* y \equiv (\exists z, w)[z \rhd^* x \wedge w \rhd^* y \wedge z \prec^* w]$

The following tree-axioms restrict all possible structures to the desired two-dimensional tree-structures described previously (and illustrated in Figure 1):

(8) There is a root that dominates all nodes:
$(\exists x)(\forall y)[x \rhd^* y \wedge \neg\exists z(z \rhd x)]$

(9) At most one parent/direct precedent per node:
$(\forall x, y, z)[[(x \prec z \wedge y \prec z) \vee (x \rhd z \wedge y \rhd z)] \rightarrow x \approx y]$

(10) Irreflexivity of $\prec$ and $\rhd$:
$(\forall x, y)[(x \prec y \vee x \rhd y) \rightarrow x \not\approx y]$

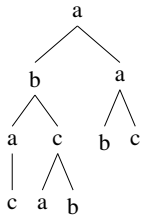
(11) Two nodes cannot be both in $\rhd^*$ and $\prec^*$ relations:
$(\forall x, y)[(x \rhd^* y \vee y \rhd^* x) \leftrightarrow \neg(x \prec^* y \vee y \prec^* x)]$

We add two final assumptions that are specific to linguistic trees. Each node can only have one label, and a node can only have a label $l \in \Sigma_{Lex}$ iff that node is a leaf (i.e. it does not dominate any other node).

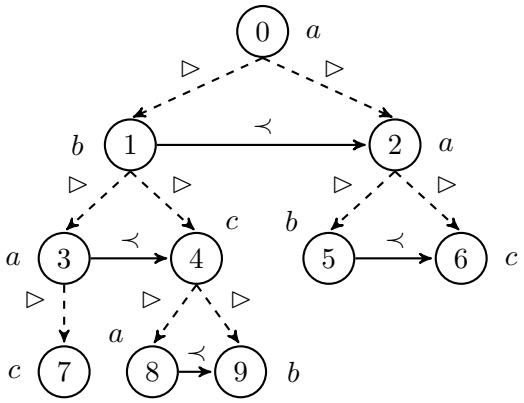
(12) $(\forall x)[(\alpha(x) \wedge \beta(x)) \rightarrow \alpha \approx \beta]$

(13) $(\forall x)[\neg\exists y(x \rhd y) \leftrightarrow \alpha(x) \wedge \alpha \in \Sigma_{Lex}]$

As an example, see tree T1 in Figure 3b, a model-theoretic representation of Figure 3a. For this non-linguistic tree, $\Sigma = \{a, b, c\}$, with no distinction between Σ_{Cat} and Σ_{Lex}. The solid lines represent the first-dimensional successor relations $\prec$, and the dashed lines represent the second-dimensional successor relations $\rhd$.



(a) Conventional representation of T1



(b) Model-theoretic representation of T1

Figure 3: T1

Then TI can be described with the following list of statements:

(14) Labeling statement:
$a(0) \wedge b(1) \wedge a(2) \wedge a(3) \wedge c(4) \wedge b(5) \wedge c(6) \wedge c(7) \wedge a(8) \wedge b(9)$

(15) Statement about $\prec$:
$1 \prec 2 \wedge 3 \prec 4 \wedge 5 \prec 6 \wedge 8 \prec 9$

(16) Statement about $\rhd$:
$0 \rhd 1 \wedge 0 \rhd 2 \wedge 1 \rhd 3 \wedge 1 \rhd 4 \wedge 2 \rhd 5 \wedge 2 \rhd 6 \wedge 3 \rhd 7 \wedge 4 \rhd 8 \wedge 4 \rhd 9$

Next, we generalize k-factors to two-dimensional trees. For strings, *k*-factors are substrings of *k* length. For trees, this will mean subtrees with depth of *k*-1 (since the depth of the root node is 0, but one node is an 1-factor of a tree).

(17) $Subtree(X) \equiv (\exists x)(\forall y)[(X(x) \wedge x \rhd^* y) \rightarrow X(y)]$

A 2-factor can be easily defined by changing the $\rhd^*$ relation to $\rhd$ in the definition of subtrees. While this is not a generalized definition of k-factors in trees, it will be sufficient for the purposes of this paper.

(18) $\text{2-Factor}(X) \equiv (\exists x)(\forall y)[(X(x) \wedge x \rhd y) \rightarrow X(y)]$

For example, the 2-factors of T1 is the set of trees in Figure 4.

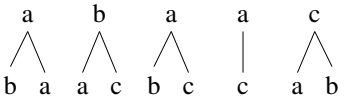


Figure 4: 2-factors of T1

In our linguistic examples, the labels will be syntactic categories (e.g. CP, NP, C', N', etc.) and language-specific lexical items. For the purposes of NPI-licensing, a specific set of lexical items are of interest only: NPIs and negation for both English and Hungarian, and CPs for Hungarian. Below we simply define the lexical items for NPIs and negation, in English and Hungarian.

(19) $\texttt{NPI}_{\texttt{eng}}(x) \equiv \text{anybody}(x) \vee \text{anything}(x) \vee \text{anywhere}(x)$

(20) $\texttt{neg}_{\texttt{eng}}(x) \equiv \text{not}(x) \vee \text{no}(x) \vee \text{nobody}(x) \vee \text{nothing}(x) \vee \text{nowhere}(x)$

(21) $\texttt{NPI}_{\texttt{hun}}(x) \equiv \text{senki}(x) \vee \text{semmi}(x) \vee \text{sehol}(x)$

(22) $\texttt{neg}_{\texttt{hun}}(x) \equiv \text{nem}(x)$

2.3 Tier-based tree-languages

Heinz et al. (2011) defined *Tier-based Strictly Local* (TSL) languages for strings. We use their definition of tiers complete with the one found in Graf and Heinz (2015) to discuss the projection of tier-trees.

A tier is denoted as $T \subseteq \Sigma$, and there is an erasing function that erases all elements in the string that are not labeled on the tier (Heinz et al., 2011).

Generalizing to our two-dimensional tree-model, a tree-tier is projected by taking only the nodes that are labeled with elements of T, while keeping all inherited left-of relations ($<^*$) and dominance relations ($\rhd^*$) between these nodes.

For example, let $T = \{a, b\}$. Applying the erasing function to T1 (3) then yields a tier-tree (5). We say that T1 *projects* a tier-tree.

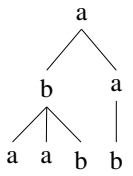
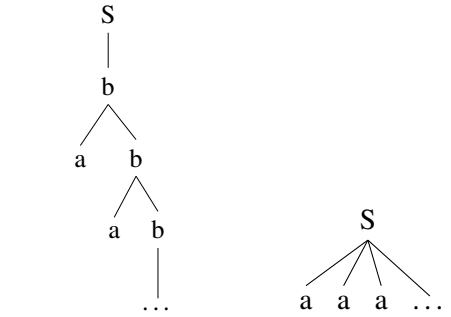


Figure 5: Tier-tree, T={a,b}

TSL string languages were characterized by a finite list of banned *k*-factors over the string-tier. The equivalent is not possible for tree-languages, because in tier-trees, there is no bound on the number of daughters for a given node.

To see why, take a tree T2 such that the starting node labeled *S* mothers a node labeled *b*, and each node *b* mothers a node *a* and *b* (6a). In linguistic terms, node *a*'s never dominate each other, neither are they ever sisters; the higher ones c-command the lower ones. Now suppose that the tier we want to project is T={S,a}. We then get a tree where *S* directly dominates an unbounded number of *a* nodes, where that number is equivalent to the depth of the original tree. We thus cannot list a finite-list of banned (or permitted) k-factors over a tree-tier without knowing a bound on the depth of the tree first.



(a) T2 (b) Tier-tree of T2, T={S,a}

Figure 6: T2 and its tier-tree

Instead, we give a general description of tree languages that are Tier-based Strictly 2-Local over the dominance relation ($TSL_2^{\rhd}$). Recall that in

the framework laid out in Rogers (2003), two-dimensional trees consist of strings (i.e. one-dimensional trees) that are dominated by a single node. We use this insight to define our tier-based grammar. Informally, the grammar contains a tier as described above, and string-based grammars that apply over sisters dominated by the same node. There are potentially as many string-based grammars as nodes labeled with a syntactic category on the tier. Formally, each grammar contains a quadruple as in Figure 7.

$$G = \langle T, T_{Cat}, H, \gamma \rangle, \text{ where:}$$

- $T \subset \Sigma$ is the finite set of tier-nodes
- $T_{Cat} = (T \cap \Sigma_{Cat})$
- H is a set of string-based grammars
- $\gamma : T_{Cat} \times H$ is a bijection that maps every node labeled $\kappa \in T_{Cat}$ to a string-based grammar $h \in H$

Figure 7: Grammar of tier-trees

The grammar defined here is thus $TSL_2^{\triangleright}$, but there can still be different types of grammar over the left-of relations. Following Graf and Heinz (2015) then, a TSL grammar for trees is a specific instance of the grammar defined here: in this case, H must be a set of TSL string-languages.

3 NPI patterns

We understand Negative Polarity Items (NPIs) as expressions that are ungrammatical in positive declarative clauses, but they are grammatical in their negative counterpart. This understanding of NPIs echoes the one for *negative dependencies* in Giannakidou and Zeijlstra (2016). For example, English *anything* is an NPI according this definition, because it shows the following contrast:

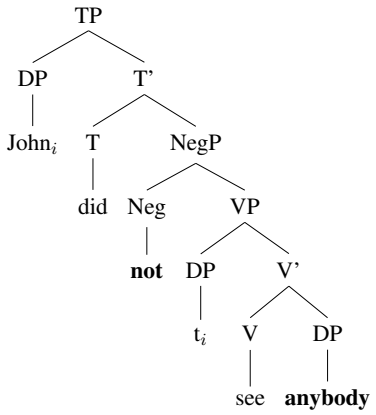
(23) a. *John has read anything.
b. John hasn't read anything.

3.1 English-type

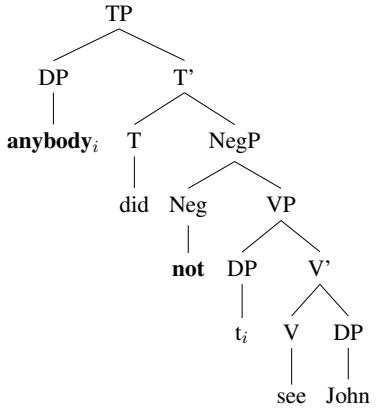
English-type NPIs are typical in English, Chinese (Lin, 1998), and Vietnamese (Tran and Bruening, 2013), among others. They are weak NPI-languages, which means that their NPIs are licensed not only by negation, but also in questions, protasis of conditionals, and in general, downward entailment contexts (Ladusaw, 1983). For the sake of simplicity, I will focus only on the cases where English NPIs are licensed by sentential negation.

The general observation is that English NPIs must be c-commanded by negation (24-26), over an arbitrary number of clause boundaries (27).[1] The NPI item *anybody* is not c-commanded by negation (*not, nobody*) in (25) and (26), but it is c-commanded and thus licensed in (24).

(24) John didn't see anybody.

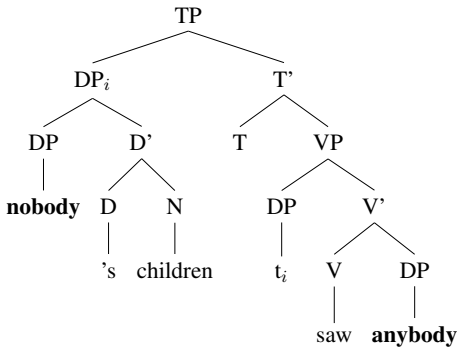


(25) *Anybody didn't see John.



(26) *Nobody's children saw anybody.

[1] A reviewer suggested that a sentence such as 'Nobody's mother understood anything I said' as a counter-evidence for the general pattern. However, if *mother* is replaced by *children*, the sentence becomes much less acceptable. Our suspicion is that 'nobody's mother' has become idiomatic, meaning 'nobody'.



(27) [John didn't think [that Charlie saw [that Mary stole anything.]]]

A straightforward way then to formalize the English-NPI licensing requirement is to state it in first-order logic with the help of defining c-command relations.

(28) $\texttt{c-com}(x, y) \equiv \neg(x \rhd^{+} y) \wedge x \not\approx y \wedge \forall z[z \rhd^{+} x \rightarrow z \rhd^{+} y]$

(29) English NPI-licensing constraint:
$\forall y[\texttt{NPI}_{\texttt{eng}}(y) \rightarrow \exists x[\texttt{c-com}(x, y) \wedge \texttt{neg}_{\texttt{eng}}(x)]]$

3.2 Hungarian-type

Here we cite our own data collected from Hungarian, but we suspect that a similar distribution is found in Slavic languages (Progovac, 1994). Hungarian NPIs show the same contrast that is found in English:

(30) *Jancsi látott senkit.
Jancsi saw NPI.ACC
'Jancsi saw anybody.'

(31) Jancsi nem látott senkit.
Jancsi NEG saw NPI.ACC
'Jancsi didn't see anybody.'

Their similarity to English NPIs stops here. Hungarian NPIs must be licensed locally by clausemate negation (32), but there is no c-command requirement for the relation between licensor and licensee (33). We assume that the domain of licensing is restricted to CP boundaries, as NPIs are licensed in sentences with negated raising predicates (34), but not in ones with negated control predicates (35).[2]

(32) *Jancsi nem tudta, hogy Mari semmit olvasott.
Jancsi NEG knew that Mari NPI.ACC read
'Jancsi didn't know that Mari read anything.'

(33) Senki nem akart el jönni.
NPI NEG want.PST PRT come.INF
'Nobody wanted to come.'

(34) Mari nem kezdett olvasni semmit.
Mari NEG started read.INF NPI.ACC
'Mari didn't start to read anything.'

(35) *Mari nem próbált olvasni semmit.
Mari NEG tried read.INF NPI.ACC
'Mari didn't try to read anything.'

The constraint can be formalized with First-order logic with the help of defining $\texttt{closest-CP}(x, y)$, which says that x is labeled CP, and it is the closest node labeled such to y.

(36) $\texttt{closest-CP}(x, y) \equiv \texttt{CP}(x) \wedge x \rhd^{*} y \wedge \neg\exists z[\texttt{CP}(z) \wedge x \rhd * + z \wedge z \rhd^{*} y]$

(37) Hungarian NPI-licensing constraint:
$\forall(y)[\texttt{NPI}_{\texttt{hun}}(y) \rightarrow \exists(x, z)[\texttt{closest-CP}(x, y) \wedge \texttt{closest-CP}(x, z) \wedge \texttt{neg}_{hun}(z)]$

3.3 Interim summary

The NPI patterns discussed above are summarized in Table 1.

	Negation must c-command NPI	Licensing across CP boundaries
English	yes	yes
Hungarian	no	no

Table 1: Summary of English and Hungarian NPI patterns.

4 Complexity of NPI patterns

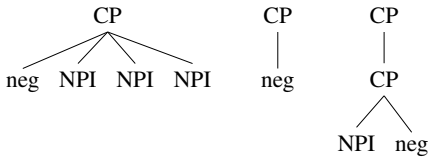
In what follows, we re-define the NPI-licensing constraints for Hungarian and English, in the context of the $\text{TSL}_2^{\rhd}$ grammar G defined in Section 2.3. We have two results: (1) Hungarian NPI-licensing can

[2] In accordance with Carnie (2013), I assume that control verbs select for CPs, whereas raising verbs select for IPs.

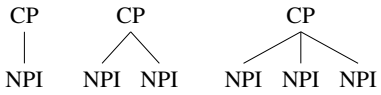
be characterized with the tier-based grammar, but English cannot, and (2) the string-language for the Hungarian grammar is neither SL or TSL, but it is LT.

4.1 Hungarian

Let us define the relevant tier for Hungarian NPI-licensing as follows: T=$\{\texttt{neg}_{hun}, \texttt{NPI}_{hun}, \texttt{CP}\}$. Then $T_{Cat} = \{\texttt{CP}\}$. For examples of grammatical tier-trees, see Figure 8.



(a) Well-formed tier-trees for Hungarian NPI-licensing



(b) Ill-formed tier-tree for Hungarian NPI-licensing

Figure 8: Well-formed and ill-formed tier-trees for Hungarian NPI-licensing

Now the question is determining the string grammar h over the nodes that CP dominates in the tier. Let us call this string-grammar h_{CP}. We show that h_{CP} is not SL, but it is LT. Consider the ill-formed set of trees where there is an arbitrary number of NPIs but there is no negation to license any of them (Figure 8b). Such trees would correspond to ungrammatical sentences of the form (38).

(38) *Senkinek^n senkije látott semmit.
NPI.DAT NPI.POSS saw NPI.ACC
'Nobody's^n anybody saw anything.'

If h_{CP} were SL, we would be able to ban a set of k-factors to successfully exclude the ill-formed trees. This is not possible for any k. For any k-factor that successfully bans a string of k-length that consists of only NPIs, there is a well-formed string of length $k+1$, whose $k+1$-th member is neg.

To exclusively define well-formed trees, the use of LT logic is necessary (39). This formula is Locally 1-Testable.

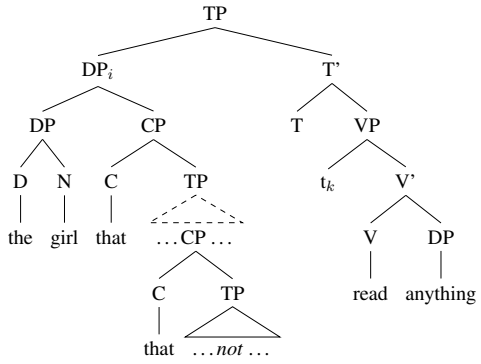
(39) $(\forall x \exists y)[\texttt{NPI}_{hun}(x) \rightarrow \texttt{neg}_{hun}(y)]$

4.2 English

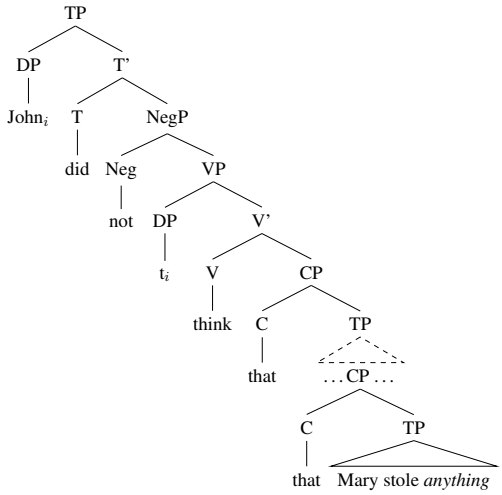
Recall that English NPI-licensing is stated as a c-command requirement: negation must c-command the NPI. This type of constraint cannot be reduced to any type of $\text{TSL}_2^{\triangleright}$ grammar.

Consider the following two sentences:

(40) *The girl, (that X said)^n that John didn't see, read anything.



(41) John didn't think (that X said)^n Mary stole anything.



In (40), negation can be found buried inside a relative clause that has been constructed through an arbitrary number of recursive embedding, and thus it does not c-command the NPI *anything*. On the other hand, in (41), negation c-commands an NPI that is buried in the embedded clauses, and thus the NPI is licensed.

There are no local constraints that can account for the restriction, since there is no bound on the distance between the negation and the NPI. Introducing tiers does not help either, due to the c-command requirement. There is no good way to define elements

for the tier to get the relevant 2-factors within the tier-tree that would help us derive the correct constraints. In fact, there is no definable tier in order to get any relevant k-factor in the tier-tree.

If the tier is defined to only include negation and NPI, the two obviously relevant elements for NPI-licensing in English, there is no way to tell apart tier-trees where negation c-commands the NPI compared to the ones where it does not. For example, both sentences (24) and (26) would yield the same tier-tree:

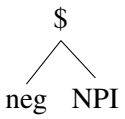


Figure 9: Tier-tree for (24) and (26), T={neg, NPI}

Including nodes that can immediately dominate negation (NegP and DP) would again result in tier-trees of arbitrary depth. For example, in sentences (41) and (40), one would have to list all the arbitrary number of DPs that serve as subjects for each embedded clause before getting to the NPI in the sentence. We then have the problem of not being able to determine subtrees of a bounded k-depth. Thus c-command relations cannot be defined using tiers.

5 Conclusions

This paper has offered four results. First, it provided a definition of $\text{TSL}_2^{\triangleright}$ languages. Second, it showed that implicational requirements, such as the surface licensing conditions of NPIs for Hungarian and English, cannot be described with TSL constraints over both dominance and precedence relations in trees.

Hungarian, which has a clausemate-requirement, can be described with a grammar that is $\text{TSL}_2^{\triangleright}$ with LT_1 constraints over the precedence relations. On the other hand, English NPI-licensing patterns, which have a structural c-command restriction, cannot be accommodated by $\text{TSL}_2^{\triangleright}$. It is yet to be seen whether English surface NPI-licensing can be described with any logical formalism that is weaker than First-order.

These results apply to surface syntactic descriptions only. Once we consider other possible theoretical explanations for NPIs that employ either feature-agreement or movement, the complexity of these syntactic constraints might be decreased. This question is to be addressed by future research.

Our class of newly defined tree-languages, $\text{TSL}_2^{\triangleright}$, needs further study also. In particular, it would be interesting to see how the characterizations of subregular string languages (e.g. the Suffix Substitution Clause for SL languages, or Local Test Invariance for LT languages) hold up once the representation changes from strings to trees. It is also yet to be seen what it means for tree languages that one can mix and match different classes of languages for different ordering relations within the same tree-structure. The nature of subregular tree-languages is still largely unknown.

Lastly, we might also examine different definitions of trees. For example, Frank and Vijay-Shanker (2001) proposed to define trees using c-command as the primitive binary relation, instead of dominance. In that case, the English NPI-licensing constraint would be very easy to state by requiring that NPIs are c-commanded by a negation. We suspect that Hungarian NPI-licensing can be accounted for as well, but a careful study is needed to confirm our hypothesis.

In conclusion, these results are only preliminary to studying the computational complexity of NPI-licensing constraints. However, they show the potential of using tools from theoretical computer science to reveal the nature of syntactic phenomena. For one, it might not be immediately obvious that on the surface level, English NPI-licensing needs more powerful tools to be described than Hungarian NPI-licensing. English is unrestricted in terms of distance between the licensor and licensee, whereas Hungarian is unrestricted in terms of structural requirements as long as the licensor and licensee are within the same clause. Formalizing these constraints using logic revealed that having an unbounded distance necessitates increased complexity compared to having no structural requirement.

Studying linguistic phenomena from a formal perspective thus can give us insight into the minimal computational requirements needed for natural language. The results in turn might bear further implications on the computational complexity needed for syntactic patterns, particularly in learning and expected cross-linguistic variation.

References

Andrew Carnie. 2013. *Syntax: A generative introduction*. John Wiley & Sons.

Jane Chandlee. 2014. *Strictly local phonological processes*. Ph.D. thesis, University of Delaware.

Herbert B Enderton. 2001. *A mathematical introduction to logic*. Academic press.

Robert Frank and K Vijay-Shanker. 2001. Primitive C-Command. *Syntax*, 4(3):164–204.

Anastasia Giannakidou and Hedde Zeijlstra. 2016. The landscape of negative dependencies: negative concord and n-words. In *Linguistics Companion*, pages 1–47. Second edition.

Thomas Graf and Jeffrey Heinz. 2015. Commonality in Disparity : The Computational View of Syntax and Phonology A New View of the Power of Syntax and Phonology.

Thomas Graf. 2013. *Local and Transderivational Constraints in Syntax and Semantics*. Ph.D. thesis, UCLA.

Thomas Graf. 2017. The Power of Locality Domains in Phonology.

Jeffrey Heinz and William Idsardi. 2013. What complexity differences reveal about domains in language. *Topics in cognitive science*, pages 111–131.

Jeffrey Heinz, Chetan Rawal, and Herbert G Tanner. 2011. Tier-based Strictly Local Constraints for Phonology. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 58–64.

Jeffrey Heinz. 2009. On the role of locality in learning stress patterns. *Phonology*, 26:303–351.

Jeffrey Heinz. 2010. Learning long-distance phonotactics. *Linguistic Inquiry*, 41:623–661.

Adam Jardine. 2016. *Locality and non-linear representations in tonal phonology*. Ph.D. thesis, University of Delaware.

William A Ladusaw. 1983. Logical Form and Conditions on Grammaticality. *Linguistics and Philosophy*, 6(3):373–392.

Jo-wang Lin. 1998. On existential polarity wh - phrases in chinese. *Journal of East Asian Linguistics*, 7(1982):219–255.

Liljiana Progovac. 1994. *Negative and Positive Polarity: A binding approach*. Cambridge University Press, Cambridge.

Geoffrey K Pullum. 2007. The Evolution of Model-Theoretic Frameworks in Linguistics. In James Rogers and Stephan Kepser, editors, *Model-Theoretic Syntax at 10*, pages 1–10, Dublin, Ireland.

James Rogers and Geoffrey K Pullum. 2011. Aural Pattern Recognition Experiments and the Subregular Hierarchy. *Journal of Logic, Language and Information*, 20(3):329–342.

James Rogers, Jeffrey Heinz, Margaret Fero, Jeremy Hurst, Dakotah Lambert, and Sean Wibel. 2013. Cognitive and Sub-regular Complexity. In *17th Conference on Formal Grammars*, pages 90–108.

James Rogers. 2003. Syntactic Structures as Multidimensional Trees. *Research on Language and Computation*, 1:265–305.

Edward P. Stabler. 1997. Derivational minimalism. *Logical aspects of computational linguistics*, pages 68–95.

Thuan Tran and Benjamin Bruening. 2013. Wh-Phrases as indefinites. A Vietnamese Perspective. In Daniel Hole and Elisabeth Löbel, editors, *Linguistics of Vietnamese: An International Survey*, pages 217–241. Mouton de Gruyter, Berlin.