Local and Long-Distance Dependencies in Syntax

Kenneth Hanson

kennethhanson.net

IACS Student Seminar Series

March 29, 2023

Introduction (1)

One of the goals of linguistics is to characterize the class of possible formal patterns: in what configurations can the elements occur?

- Phonology: how can the sounds that make up words be arranged?
- Syntax: how can the words that make up sentences be arranged?

It is common to model phonological patterns with strings (sequences), and syntactic patterns with trees.

Introduction (2)

Recent work hypothesizes that local dependencies in language are in the formal class **strictly local (SL)**, while long-distance dependencies are **tier-based strictly local (TSL)** (Heinz 2018; Graf 2022a).

The hypothesis is well supported for phonology, but confirming it for syntax is an area of ongoing research.

Overview

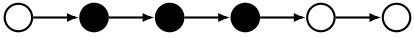
Examples of local and long-distance dependencies in linguistics:

Patterns in	Modeled As	Local	Long-Distance
Phonology	Strings	Syllable Structure	Consonant Harmony
Syntax	Trees	Category Selection	**Agreement (New!)**

Outline

- Local string patterns
- Long-distance string patterns
- From strings to trees
- Local dependencies in syntax
- Long-distance dependencies in syntax

Local String Patterns



Example: Syllable Structure

Some possible and impossible sound sequences in English:

CVC syllable:	tap, kap, kaʃ, …
Cr at start of word:	trap, kraʃ, …
No rC at start of word:	*rtap, *rkaʃ, …
rC at the end of word:	tarp, ʃark, …

C stands for any consonant other than ‘r’. V stands for any vowel. ʃ = the ‘sh’ sound. * = unacceptable.

Modeling Local Patterns

L = {tap, kaʃ, trap, kraʃ, tarp, ʃark, . . . } $\bar{L}$ = {*rtap, *kaʃ, . . . }

We can determine whether a string is grammatical by looking at substrings of a fixed length.

$ ʃ a r k $

$ marks the beginning and end of a word.

Modeling Local Patterns

L = {tap, kaʃ, trap, kraʃ, tarp, ʃark, . . . } $\bar{L}$ = {*rtap, *kaʃ, . . . }

We can determine whether a string is grammatical by looking at substrings of a fixed length.

$ marks the beginning and end of a word.

Modeling Local Patterns

L = {tap, kaʃ, trap, kraʃ, tarp, ʃark, . . . } $\bar{L}$ = {*rtap, *kaʃ, . . . }

We can determine whether a string is grammatical by looking at substrings of a fixed length.

$ marks the beginning and end of a word.

Modeling Local Patterns

L = {tap, kaʃ, trap, kraʃ, tarp, ʃark, . . . } $\bar{L}$ = {*rtap, *kaʃ, . . . }

We can determine whether a string is grammatical by looking at substrings of a fixed length.

$ marks the beginning and end of a word.

Modeling Local Patterns

L = {tap, kaʃ, trap, kraʃ, tarp, ʃark, . . . } $\bar{L}$ = {*rtap, *kaʃ, . . . }

We can determine whether a string is grammatical by looking at substrings of a fixed length.

$ marks the beginning and end of a word.

Modeling Local Patterns

L = {tap, kaʃ, trap, kraʃ, tarp, ʃark, . . . } $\bar{L}$ = {*rtap, *kaʃ, . . . }

We can determine whether a string is grammatical by looking at substrings of a fixed length.

$ ʃ a r k $ ✓

$ marks the beginning and end of a word.

Modeling Local Patterns

L = {tap, kaʃ, trap, kraʃ, tarp, ʃark, . . . } $\bar{L}$ = {*rtap, *kaʃ, . . . }

We can determine whether a string is grammatical by looking at substrings of a fixed length.

$ ʃ a r k $ ✓

$ k r a ʃ $

$ marks the beginning and end of a word.

Modeling Local Patterns

L = {tap, kaʃ, trap, kraʃ, tarp, ʃark, . . . } $\bar{L}$ = {*rtap, *kaʃ, . . . }

We can determine whether a string is grammatical by looking at substrings of a fixed length.

$	ʃ	a	r	k	$	✓
$	k	r	a	ʃ	$	

$ marks the beginning and end of a word.

Modeling Local Patterns

L = {tap, kaʃ, trap, kraʃ, tarp, ʃark, . . . } $\bar{L}$ = {*rtap, *kaʃ, . . . }

We can determine whether a string is grammatical by looking at substrings of a fixed length.

$	ʃ	a	r	k	$	✓
$	k	r	a	ʃ	$	

$ marks the beginning and end of a word.

Modeling Local Patterns

L = {tap, kaʃ, trap, kraʃ, tarp, ʃark, . . . } $\bar{L}$ = {*rtap, *kaʃ, . . . }

We can determine whether a string is grammatical by looking at substrings of a fixed length.

$ ʃ a r k $ ✓

$ k r a ʃ $

$ marks the beginning and end of a word.

Modeling Local Patterns

L = {tap, kaʃ, trap, kraʃ, tarp, ʃark, . . . } $\bar{L}$ = {*rtap, *kaʃ, . . . }

We can determine whether a string is grammatical by looking at substrings of a fixed length.

$	ʃ	a	r	k	$	✓
$	k	r	a	ʃ	$	

$ marks the beginning and end of a word.

Modeling Local Patterns

L = {tap, kaʃ, trap, kraʃ, tarp, ʃark, . . . } $\bar{L}$ = {*rtap, *kaʃ, . . . }

We can determine whether a string is grammatical by looking at substrings of a fixed length.

$	ʃ	a	r	k	$	✓
$	k	r	a	ʃ	$	✓

$ marks the beginning and end of a word.

Modeling Local Patterns

L = {tap, kaʃ, trap, kraʃ, tarp, ʃark, …} $\bar{L}$ = {*rtap, *kaʃ, …}

We can determine whether a string is grammatical by looking at substrings of a fixed length.

$	ʃ	a	r	k	$	✓
$	k	r	a	ʃ	$	✓
$	r	k	a	ʃ	$	

$ marks the beginning and end of a word.

Modeling Local Patterns

L = {tap, kaʃ, trap, kraʃ, tarp, ʃark, . . . } $\bar{L}$ = {*rtap, *kaʃ, . . . }

We can determine whether a string is grammatical by looking at substrings of a fixed length.

$	ʃ	a	r	k	$	✓
$	k	r	a	ʃ	$	✓
$	r	k	a	ʃ	$	

$ marks the beginning and end of a word.

Modeling Local Patterns

L = {tap, kaʃ, trap, kraʃ, tarp, ʃark, . . . } $\bar{L}$ = {*rtap, *kaʃ, . . . }

We can determine whether a string is grammatical by looking at substrings of a fixed length.

$	ʃ	a	r	k	$	✓
$	k	r	a	ʃ	$	✓
$	r	k	a	ʃ	$	

$ marks the beginning and end of a word.

Modeling Local Patterns

L = {tap, kaʃ, trap, kraʃ, tarp, ʃark, . . . } $\bar{L}$ = {*rtap, *kaʃ, . . . }

We can determine whether a string is grammatical by looking at substrings of a fixed length.

$	ʃ	a	r	k	$	✓
$	k	r	a	ʃ	$	✓
$	r	k	a	ʃ	$	

$ marks the beginning and end of a word.

Modeling Local Patterns

L = {tap, kaʃ, trap, kraʃ, tarp, ʃark, . . . } $\bar{L}$ = {*rtap, *kaʃ, . . . }

We can determine whether a string is grammatical by looking at substrings of a fixed length.

$	ʃ	a	r	k	$	✓
$	k	r	a	ʃ	$	✓
$	r	k	a	ʃ	$	

$ marks the beginning and end of a word.

Modeling Local Patterns

L = {tap, kaʃ, trap, kraʃ, tarp, ʃark, . . . } $\bar{L}$ = {*rtap, *kaʃ, . . . }

We can determine whether a string is grammatical by looking at substrings of a fixed length.

$	ʃ	a	r	k	$	✓
$	k	r	a	ʃ	$	✓
$	r	k	a	ʃ	$	✗

$ marks the beginning and end of a word.

Strictly Local Languages

L = {tap, kaʃ, trap, kraʃ, tarp, ʃark, . . . } $\bar{L}$ = {*rtap, *kaʃ, . . . }

In a strictly local (SL) language, a string is well-formed iff all of its substrings of a fixed length (k) are well-formed.

An SL grammar is just a set of valid substrings. For our English example:

k = 3

G = {$CV, $Cr, CrV, CVC, CVr, rVC, VrC, VC$, rC$}

$\bar{G}$ = {$CC, $rC, C, . . . }

‘C’ stands for any consonant other than ‘r’. ‘V’ stands for any vowel.

Extending the Model

L = {tap, kaʃ, trap, kraʃ, tarp, ʃark, . . . } $\bar{L}$ = {*rtap, *kaʃ, . . . }

k = 3

G = {$CV, $Cr, CrV, CVC, CVr, rVC, VrC, VC$, rC$}

$\bar{G}$ = {$CC, $rC, C, . . . }

- A window size of 3 is sufficient for most of English.
- To allow multiple syllables, just add {VCV, rCV, VCC, CCV} to the grammar.
- To make finer distinctions, split C and V into smaller sets of symbols.

Computational Complexity

Strictly local languages are extremely restricted in expressive power. They lie at the very bottom of the hierarchy of formal languages (McNaughton and Papert 1971).

1. The patterns they can encode are extremely restricted.
2. They are efficient to process.
3. They are easy to learn, though you need to guess the window size k.

Computational Complexity

1. The patterns that SL languages encode are extremely restricted.
   - No relationship between symbols at arbitrary distance
   - No boolean conditions
   - No counting of substrings

Computational Complexity

2. SL languages are efficient to process.

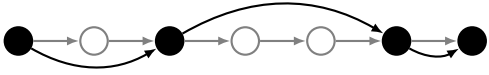
- Size of the grammar is at most $|\Sigma|^k$, where Σ is the set of symbols.
- Testing or generating a string takes linear time, e.g. when implemented as a finite state machine.

Computational Complexity

3. SL languages are easy to learn.

- Just keep track of all attested substrings of size *k*
  → *string extension learning* (Garcia et al. 1990; Heinz 2010).
- The required computations are cognitively plausible (Lambert et al. 2021).
- Time to process data is linear.
- Very little data is needed (compared to more expressive classes).

Long-Distance String Patterns



Example: Consonant Harmony

Example from Samala (extinct, Chumash family, southern California)

Sibilants (s, ʃ) must match, no matter the distance.

Possible words	Impossible words
ʃtojonowonowaʃ	stojonowonowaʃ
stojonowonowas	ʃtojonowonowas
pisotonosikiwat	pisotonoʃikiwat
nasipisotonosikiwa	naʃipisotonoʃikiwa

This example is from Heinz (2018).

How to Model

If we ignore all the irrelevant material, we can enforce matching with a window of size 2.

Possible words		Impossible words	
ʃtojonowonowaʃ	ʃʃ	stojonowonowaʃ	sʃ
stojonowonowas	ss	ʃtojonowonowas	ʃs
pisotonosikiwat	ss	pisotonoʃikiwat	sʃ
nasipisotonosikiwa	sss	naʃipisotonoʃikiwa	ʃsʃ

Tier-Based Strictly Local Languages

SL languages are defined in terms of strings of adjacent symbols.

Tier-based strictly local (TSL) languages make use of relativized adjacency — what is adjacent when the irrelevant material is removed.

Define a tier T of salient symbols and enforce an SL grammar.
For Samala consonant harmony:

$T = \{\text{s}, \text{ʃ}\}$ $k = 2$ $G = \{\$\text{s}, \$\text{ʃ}, \text{ss}, \text{ʃʃ}, \text{s}\$, \text{ʃ}\$\}$ $\bar{G} = \{\text{sʃ}, \text{ʃs}\}$

See Heinz et al. (2011) for details.

Some Definitions

SL language: $w \in L \Leftrightarrow \text{substrings}_k(w) \subseteq G$

TSL language: $w \in L \Leftrightarrow \text{substrings}_k(\text{project}_T(w)) \subseteq G$

$w \in \Sigma^*, L \subseteq \Sigma^*, T \subseteq \Sigma, G \subseteq \Sigma^k$

Computational Complexity, Revisited

TSL may enforce long-distance dependencies, but it is otherwise limited in the same way as SL.

- Still no boolean conditions, no counting of substrings.
- Memory and runtime complexity are still low.
- Learning is still easy, though you need to know/guess the tiers (Lambert 2021).

Interim Summary

SL and TSL are models of local and long-distance dependencies, respectively.

- They are conceptually and computationally simple.
- Their mathematics is well-understood.

Question: Are they empirically adequate?

Empirical Adequacy

Hypothesis: **all** local dependencies in linguistics are SL, while long-distance dependencies are TSL (or a slight variant thereof).

For phonology, the hypothesis has good support (Heinz 2018).

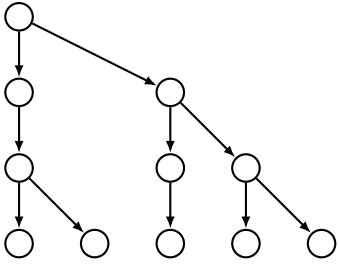
For syntax, the tree-based equivalents of these classes seem to be a good model (Graf 2022a). This is work in progress.

Why SL and TSL Matter

If correct, the hypothesis is informative both from a scientific and engineering perspective.

- Science: it helps to explain linguistic typology and link linguistic theory to broader issues in cognitive science.
- Engineering: we can use these models in engineering applications — and not as a compromise!

From Strings to Trees



The Need for Trees (1)

Syntax has hierarchical structure — groups of words behave as a unit.

Examples:

- Pronominalization
  - [The president of the United States] has arrived.
  - **He** has arrived.
  - **Who** has arrived?
- Movement/displacement
  - I like [this one].
  - [This one], I like.

The Need for Trees (2)

Long-distance dependencies are rampant:

[This cat] sleeps under the bed.

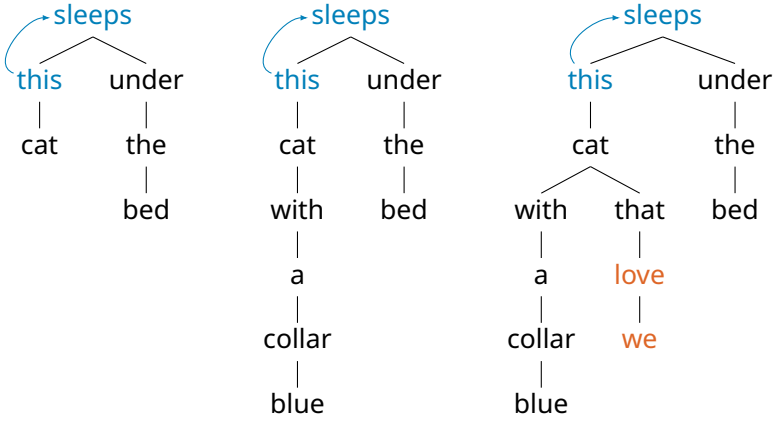
[This cat [with a blue collar]] sleeps under the bed.

And we can't simply ignore only elements of the wrong type:

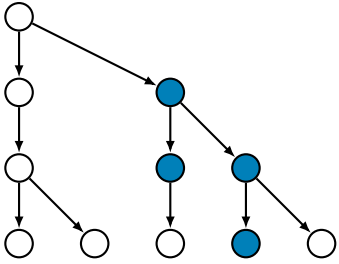
[This cat [with a blue collar] [that we love]] sleeps under the bed.

The Need for Trees (3)

Interestingly, many (but not all!) long-distance dependencies become local once we switch to a hierarchical representation.

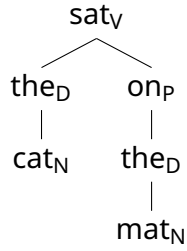


Local Dependencies in Syntax



Phrase Structure

The cat sat on the mat.

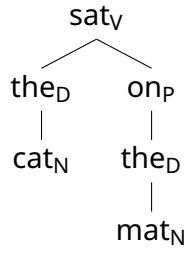


- Each node has a category: N(oun), V(erb), P(reposition), D(eterminer), etc.
- Each **subtree** (a node and its descendants) represents a phrase.
- The category of a phrase is determined by the root of the subtree.

These trees are dependency trees. See (Graf 2022b).

Category Selection

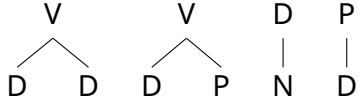
The cat sat on the mat.



- Verb *sat* selects a DP subject and PP object.
- Determiner *the* selects a NP.
- Preposition *on* selects a DP.

How to Model (1)

One approach: define a set of “treelets” that combine to make bigger trees (Rogers 1997).



This works well for local patterns, but is awkward to generalize to long-distance patterns.

How to Model (2)

Another approach: extract strings from the trees which contain the relevant (hierarchical) information, then apply an SL (or TSL) grammar as before.

For category selection, we want strings which contain a node and its children within a fixed window.

Then we can define an SL grammar such as the following:

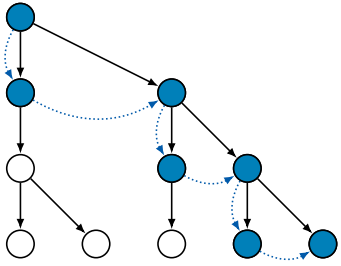
$$G = \{ \mathrm{V} \cdot \mathrm{D} \cdot \mathrm{D},\ \ \mathrm{V} \cdot \mathrm{D} \cdot \mathrm{P},\ \ \mathrm{D} \cdot \mathrm{N},\ \ \mathrm{P} \cdot \mathrm{D},\ \ \ldots \}$$

Assuming an SL grammar with window size k, a substring u of length $j < k$ is a shorthand for all substrings v of length k such that u is a suffix of v.

Command Strings

The **command string (c-string)** of a node is a hierarchical ordering of its ancestors and their left siblings, such that sibling order is preserved.

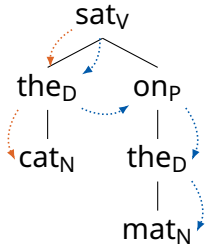
The c-string of node contains the necessary information to enforce category selection for that node and all of its ancestors.



See Graf and Shafiei (2019) for details on c-strings.

See Graf and De Santo (2019) for how c-string grammars can be compiled into a tree grammar.

Category Selection with C-Strings



$$G = \left\{ \begin{array}{c} V \cdot D \cdot D \\ V \cdot D \cdot P \\ D \cdot N \\ P \cdot D \\ \vdots \end{array} \right\}$$

C-string for *mat*: $\text{sat}_V \cdot \text{the}_D \cdot \text{on}_P \cdot \text{the}_D \cdot \text{mat}_N$

C-string for *cat*: $\text{sat}_V \cdot \text{the}_D \cdot \text{cat}_N$

Is anything else in syntax SL?

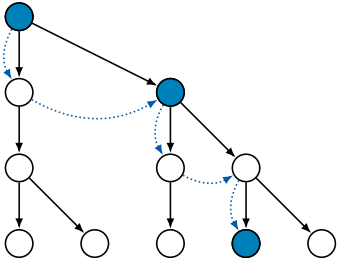
Yes! **Functional hierarchies** are also SL.

Example: English clausal auxiliaries

	Modal	Perfect	Progressive	V	
John				eats	ice cream.
John	will			eat	ice cream.
John		has		eaten	ice cream.
John			is	eating	ice cream.
John	will	have		eaten	ice cream.
John	will		be	eating	ice cream.
John		has	been	eating	ice cream.
John	will	have	been	eating	ice cream.

$$G = \left\{ \begin{array}{llll} \$\ \text{Mod} & & & \\ \$\ \text{Perf} & \text{Mod Perf} & & \\ \$\ \text{Prog} & \text{Mod Prog} & \text{Perf Prog} & \\ \$\ \text{V} & \text{Mod V} & \text{Perf V} & \text{Prog V} \end{array} \right\}$$

Long-Distance Dependencies in Syntax



Agreement (1)

Agreement: certain elements in the sentence much “match” in some feature, such as grammatical number.

For example, in many languages (including English), the verb must agree with the subject (in a simple sentence). It cannot agree with the object.

✓ This cat chases those rats.

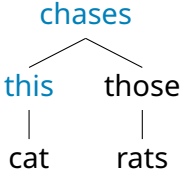
✗ This cat chase those rats.

Agreement (2)

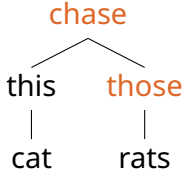
In a simple sentence, the verb must agree with the subject.

It cannot agree with the object.

✓ This cat chases those rats.



✗ This cat chase those rats.



Agreement at a Distance (1)

Agreement is often local, but it can also operate at a distance.

In English, this happens when the subject is expletive (pleonastic) *there*.

There seems to be a lion in the garden.

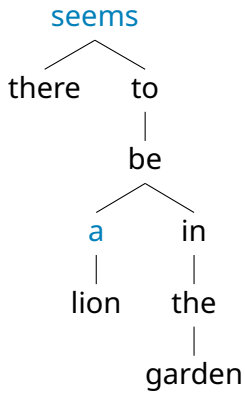
There seem to be some lions in the garden.

But there are still limits. Agreement must target the closest suitable DP.

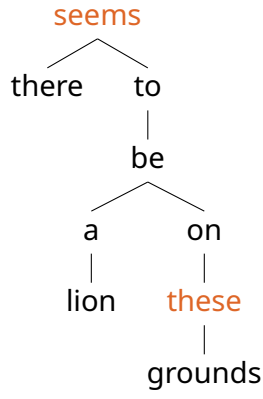
✗ There seem to be a lion on these grounds.

Agreement at a Distance (2)

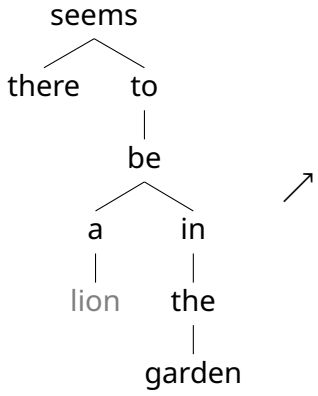
✓ There seems to be a lion in the garden.



✗ There seem to be a lion on these grounds.



Agreement with C-Strings (1)



seems · there · to · be · a · in · the · garden

↓

seems · a · the

Agreement with C-Strings (2)

There seems to be a lion in the garden.

↓

seems · there · to · be · a · in · the · garden

T = {all V, all D}
k = 2
G = {V_{SG} D_{SG}, V_{PL} D_{PL}, …}
$\bar{G}$ = {V_{SG} D_{PL}, V_{PL} D_{SG}, …}

Invisibility and Blocking

Recall that the core property of TSL is relativized adjacency.

This mathematical property derives two empirical properties of long-distance dependencies:

- Invisibility – elements which are inactive
- Blocking – elements which interfere with dependency formation

In the above example, elements other than V and D are invisible.

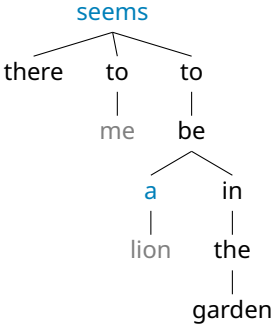
What about blockers?

Dative Intervention (1)

In many languages, DPs marked with **dative case** do not participate in agreement, but block agreement when they intervene.

In English, this doesn't happen with expletive *there*:

✓ There seem **to me** to be some lions in the garden.



→ seems · there · to · to · be · some · in · the · garden

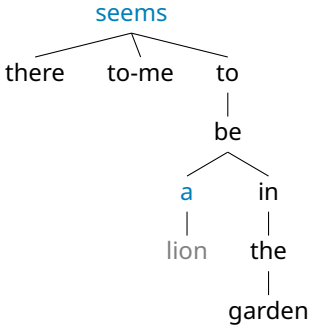
Dative Intervention (2)

Now, imagine a language with *dative case*.

In addition to *I*/*me*/*my*, it also has a dative pronoun *to-me*.

In such languages, datives often block agreement:

✗ There seem **to-me** to be some lions in the garden.



→ seems · there · **to-me** · to · be · some · in · the · garden

The Typology of Agreement

If we vary the tier projection and constraints slightly, we can account for cross-linguistic variation in agreement patterns.

Case studies conducted so far (Hanson n.d.):

- Case-sensitive agreement (Hindi, Mahajan 1990)
- Dative intervention (Icelandic, Holmberg and Hróarsdóttir 2003)
- Upward agreement in (Lubukusu, Diercks 2013)
- A-bar agreement in (Dinka, Van Urk 2015)
- Agreement across a finite clause (Zulu, Halpert 2019)

Conclusion

Local patterns in linguistics are SL, and long-distance patterns are TSL.

In phonology, these are string patterns.

In syntax, these are c-strings over trees.

What's Next?

Work that remains to be done:

- Recast previous work on long-distance syntactic dependencies in the current framework.
- Survey a wider range of languages to strengthen support for the hypothesis.
- Develop a learning model for syntactic patterns based on these results.

Thank you!

Acknowledgments

- This work was supported by NSF Grant BCS-1845344.

References

Diercks, Michael (2013). “Indirect agree in Lubukusu complementizer agreement”. In: *Natural Language & Linguistic Theory* 31.2.

Garcia, Pedro et al. (1990). “Learning Locally Testable Languages in the Strict Sense”. In: *Proceedings of the Workshop on Algorithmic Learning Theory*.

Graf, Thomas (2022a). “Subregular linguistics: bridging theoretical linguistics and formal grammar”. In: *Theoretical Linguistics* 48.3-4. DOI: doi:10.1515/tl-2022-2037.

— (2022b). “Typological implications of tier-based strictly local movement”. In: *Proceedings of the Society for Computation in Linguistics 2022*.

Graf, Thomas and Aniello De Santo (2019). “Sensing Tree Automata as a Model of Syntactic Dependencies”. In: *Proceedings of the 16th Meeting on the Mathematics of Language*. Toronto, Canada: Association for Computational Linguistics.

Graf, Thomas and Nazila Shafiei (2019). “C-command dependencies as TSL string constraints”. In: *Proceedings of the Society for Computation in Linguistics (SCiL) 2019*.

Halpert, Claire (2019). “Raising, unphased”. In: *Natural Language & Linguistic Theory* 37.1.

Hanson, Kenneth (n.d.). *A Computational Perspective on the Typology of Agreement*.

Heinz, Jeffrey (July 2010). “String Extension Learning”. In: *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Uppsala, Sweden: Association for Computational Linguistics.

— (2018). “The computational nature of phonological generalizations”. In: *Phonological Typology, Phonetics and Phonology*.

References (2)

Heinz, Jeffrey et al. (2011). “Tier-based strictly local constraints for phonology”. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human language technologies*.

Holmberg, Anders and Thorbjörg Hróarsdóttir (2003). “Agreement and movement in Icelandic raising constructions”. In: *Lingua* 113.10. ISSN: 0024-3841. DOI: https://doi.org/10.1016/S0024-3841(02)00162-6.

Lambert, Dakotah (2021). “Grammar Interpretations and Learning TSL Online”. In: *Proceedings of the Fifteenth International Conference on Grammatical Inference*. Vol. 153. Proceedings of Machine Learning Research. PMLR.

Lambert, Dakotah et al. (2021). “Typology emerges from simplicity in representations and learning”. In: *Journal of Language Modelling* 9.1.

Mahajan, Anoop Kumar (1990). “The A/A-bar distinction and movement theory”. PhD thesis. Massachusetts Institute of Technology.

McNaughton, Robert and Seymour A. Papert (1971). *Counter-Free Automata*. The MIT Press.

Rogers, James (1997). “Strict LT_2 : Regular :: Local : Recognizable”. In: *Logical Aspects of Computational Linguistics: First International Conference, LACL ’96 (Selected Papers)*. Vol. 1328. Lectures Notes in Computer Science/Lectures Notes in Artificial Intelligence. Springer.

Van Urk, Coppe (2015). “A Uniform Syntax for Phrasal Movement: A Dinka Bor Case Study”. PhD thesis. Cambridge, Mass.: MIT.