The (Surprising) Simplicity of Syntax

Derivation trees, subregular complexity, and what it implies for language and cognition

Thomas Graf

Stony Brook University
mail@thomasgraf.net
http://thomasgraf.net

Penn ILST Seminar
November 26, 2018

You can get the slides here under "News"

A Mathematical Distinctness Theorem

- From a computational perspective,
  there is a split between "P-side" and "S-side".

regular $<$ context-free $<$ mildly context-sensitive $< \cdots$

Phonology

Morphology

Syntax

- Matches linguistic practice
  (despite attempts at unification, e.g. DM)
- Why is syntax the outlier?

A Mathematical Distinctness Theorem

- From a computational perspective, there is a split between "P-side" and "S-side".

regular $<$ context-free $<$ mildly context-sensitive $< \cdots$

Kaplan and Kay (1994)

Phonology

Morphology

Syntax

- Matches linguistic practice (despite attempts at unification, e.g. DM)
- Why is syntax the outlier?

A Mathematical Distinctness Theorem

- From a computational perspective,
  there is a split between "P-side" and "S-side".

regular < context-free < mildly context-sensitive < $\cdots$

Kaplan and Kay (1994)

Phonology

Karttunen et al. (1992)

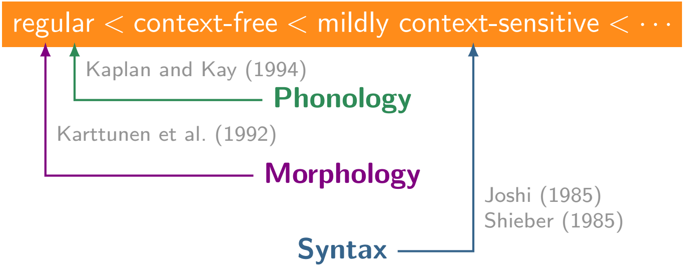
Morphology

Syntax

- Matches linguistic practice
  (despite attempts at unification, e.g. DM)
- Why is syntax the outlier?

A Mathematical Distinctness Theorem

- From a computational perspective, there is a split between "P-side" and "S-side".



- Matches linguistic practice (despite attempts at unification, e.g. DM)
- Why is syntax the outlier?

An Alternative: Simple Syntax

- The postulated split is misleading.
- If we probe deeper, we find that
  - syntax is just as simple,
  - phonology, morphology, and syntax are weaker than regular $\Rightarrow$ **subregular**
  - relativized locality plays a major role,
  - and is approximated by the formal class **TSL**.
- This has repercussions for
  - cognitive architecture of language,
  - learning,
  - processing.

Outline

The Subregular Program

- **Received view**: class of regular (= finite-state) string languages maximally complex
- **Subregular hierarchy**: even weaker/simpler subclasses
- The **tier-based strictly local (TSL)** languages have emerged as particularly important.

Jeff Heinz

Jane Chandlee

Adam Jardine

Kevin McMullin

Example: Word-Final Devoicing

- Captured by forbidding voiced segments at the end of a word
- **German**: Don't have **z$** or **v$** or **d$** (where **$** = word edge).

Example: German

	***z$**
* **$** r a **d** **$**	***v$**
	***d$**

Example: Word-Final Devoicing

- Captured by forbidding voiced segments at the end of a word
- **German**: Don't have **z$** or **v$** or **d$** (where **$** = word edge).

Example: German

* **$** r a **d** **$**

***z$**
***v$**
***d$**

Example: Word-Final Devoicing

- Captured by forbidding voiced segments at the end of a word
- **German**: Don't have **z$** or **v$** or **d$** (where **$** = word edge).

Example: German

* $\boxed{\$\ r}$ a **d** **$**

***z$**

***v$**

***d$**

Example: Word-Final Devoicing

- Captured by forbidding voiced segments at the end of a word
- **German**: Don't have **z$** or **v$** or **d$** (where **$** = word edge).

Example: German

* **$** $\boxed{\text{r a}}$ **d** **$**

***z$**

***v$**

***d$**

Example: Word-Final Devoicing

- Captured by forbidding voiced segments at the end of a word
- **German**: Don't have **z$** or **v$** or **d$** (where **$** = word edge).

Example: German

* **$** r a **d** **$**

***z$**

***v$**

***d$**

Example: Word-Final Devoicing

- Captured by forbidding voiced segments at the end of a word
- **German**: Don't have **z$** or **v$** or **d$** (where **$** = word edge).

Example: German

* **$** r a **d $**

*__z$__

*__v$__

*__d$__

Example: Word-Final Devoicing

- Captured by forbidding voiced segments at the end of a word
- **German**: Don't have **z$** or **v$** or **d$** (where **$** = word edge).

Example: German

* $ r a **d $**

*z$

*v$

*d$

Example: Word-Final Devoicing

- Captured by forbidding voiced segments at the end of a word
- **German**: Don't have **z$** or **v$** or **d$** (where **$** = word edge).

Example: German

	***z$**	
* **$** r a **d** **$**	***v$**	**$** r a t **$**
	***d$**	

Example: Word-Final Devoicing

- Captured by forbidding voiced segments at the end of a word
- **German**: Don't have **z$** or **v$** or **d$** (where **$** = word edge).

Example: German

* $ r a d $

*z$
*v$
*d$

$ r a t $

Example: Word-Final Devoicing

- Captured by forbidding voiced segments at the end of a word
- **German**: Don't have **z$** or **v$** or **d$** (where **$** = word edge).

Example: German

* $ r a d $

*z$
*v$
*d$

$ $\boxed{\text{r a}}$ t $

Example: Word-Final Devoicing

- Captured by forbidding voiced segments at the end of a word
- **German**: Don't have **z$** or **v$** or **d$** (where **$** = word edge).

Example: German

*\$ r a d \$

*z\$

*v\$

*d\$

\$ r $\boxed{\text{a t}}$ \$

Example: Word-Final Devoicing

- Captured by forbidding voiced segments at the end of a word
- **German**: Don't have **z$** or **v$** or **d$** (where **$** = word edge).

Example: German

* **$** r a **d** **$**   ***z$**
 ***v$**
 ***d$**   **$** r a t **$**

Example: Intervocalic Voicing

- Captured by forbidding voiceless segments between vowels
- **Suppose**:
  - $[-\text{voice}] = \{$s,ʃ$\}$
  - $\text{V} = \{$a,i,u$\}$
- **Then**: don't have **asa**, **aʃa**, **asi**, **aʃi**, . . .

Example

* $ a z u s a $

Example: Intervocalic Voicing

- Captured by forbidding voiceless segments between vowels
- **Suppose**:
  - $[-\text{voice}] = \{\text{s,ʃ}\}$
  - $\text{V} = \{\text{a,i,u}\}$
- **Then**: don't have **asa**, **aʃa**, **asi**, **aʃi**, ...

Example

* $ **a** z **u** **s** **a** $

Example: Intervocalic Voicing

- Captured by forbidding voiceless segments between vowels
- **Suppose**:
  - $[-\text{voice}] = \{$s,ʃ$\}$
  - $\mathrm{V} = \{$a,i,u$\}$
- **Then**: don't have **asa**, **aʃa**, **asi**, **aʃi**, . . .

Example

*$ a z u s a $

Example: Intervocalic Voicing

- Captured by forbidding voiceless segments between vowels
- **Suppose**:
  - $[-\text{voice}] = \{$s,ʃ$\}$
  - $\text{V} = \{$a,i,u$\}$
- **Then**: don't have **asa**, **aʃa**, **asi**, **aʃi**, . . .

Example

* \$ $\boxed{\textbf{a}\ \text{z}\ \textbf{u}}$ **s a** \$

Example: Intervocalic Voicing

- Captured by forbidding voiceless segments between vowels
- **Suppose**:
  - $[-\text{voice}] = \{$s,ʃ$\}$
  - $\mathrm{V} = \{$a,i,u$\}$
- **Then**: don't have **asa**, **aʃa**, **asi**, **aʃi**, . . .

Example

* \$ **a** $\boxed{\text{z } \textbf{u} \textbf{ s}}$ **a** \$

Example: Intervocalic Voicing

- Captured by forbidding voiceless segments between vowels
- **Suppose**:
  - $[-\text{voice}] = \{$s,ʃ$\}$
  - $\text{V} = \{$a,i,u$\}$
- **Then**: don't have **asa**, **aʃa**, **asi**, **aʃi**, . . .

Example

* \$ **a** z $\boxed{\textbf{u s a}}$ \$

A Problem: Samala Sibilant Harmony

- If multiple sibilants occur in the same word, they must all be +anterior (**s**,**z**) or −anterior (**ʃ**,**ʒ**).
- In other words: Don't mix **purple** and **teal**.

*sʃ *sʒ *zʃ *zʒ
*ʃs *ʒs *ʃz *ʒz

- **But:** Sibilants can be arbitrarily far away from each other!

Example: Samala

*$ h a s x i n t i l a w a ʃ $

$ h a ʃ x i n t i l a w a ʃ $

A Problem: Samala Sibilant Harmony

- If multiple sibilants occur in the same word, they must all be +anterior (**s**,**z**) or −anterior (**ʃ**,**ʒ**).
- In other words: Don't mix **purple** and **teal**.

*sʃ *sʒ *zʃ *zʒ
*ʃs *ʒs *ʃz *ʒz

- **But:** Sibilants can be arbitrarily far away from each other!

Example: Samala

*$ h a **s** x i n t i l a w a **ʃ** $

$ h a **ʃ** x i n t i l a w a **ʃ** $

A Problem: Samala Sibilant Harmony

- If multiple sibilants occur in the same word, they must all be +anterior (**s**,**z**) or −anterior (**ʃ**,**ʒ**).
- In other words: Don't mix **purple** and **teal**.

*sʃ *sʒ *zʃ *zʒ
*ʃs *ʒs *ʃz *ʒz

- **But:** Sibilants can be arbitrarily far away from each other!

Example: Samala

*$ h a s x i n t i l a w a ʃ $

$ h a ʃ x i n t i l a w a ʃ $

A Problem: Samala Sibilant Harmony

- If multiple sibilants occur in the same word, they must all be +anterior (s,z) or −anterior (ʃ,ʒ).
- In other words: Don't mix **purple** and **teal**.

*sʃ	*sʒ	*zʃ	*zʒ
*ʃs	*ʒs	*ʃz	*ʒz

- **But:** Sibilants can be arbitrarily far away from each other!

Example: Samala

*\$ h a $\boxed{\text{s x i n t i l a w a ʃ}}$ \$

\$ h a $\boxed{\text{ʃ x i n t i l a w a ʃ}}$ \$

A Problem: Samala Sibilant Harmony

- If multiple sibilants occur in the same word, they must all be +anterior (**s**,**z**) or −anterior (**ʃ**,**ʒ**).
- In other words: Don't mix **purple** and **teal**.

*sʃ *sʒ *zʃ *zʒ
*ʃs *ʒs *ʃz *ʒz

- **But:** Sibilants can be arbitrarily far away from each other!

Example: Samala

*$ h a s x i n t i l a w a ʃ $

$ h a ʃ x i n t i l a w a ʃ $

*$ s t a j a n o w o n w a ʃ $

A Problem: Samala Sibilant Harmony

- If multiple sibilants occur in the same word, they must all be +anterior (s,z) or −anterior (ʃ,ʒ).
- In other words: Don't mix **purple** and **teal**.

*sʃ *sʒ *zʃ *zʒ
*ʃs *ʒs *ʃz *ʒz

- **But:** Sibilants can be arbitrarily far away from each other!

Example: Samala

*$ h a s x i n t i l a w a ʃ $

$ h a ʃ x i n t i l a w a ʃ $

*$ s t a j a n o w o n w a ʃ $

Making Long-Distance Dependencies Local

- Let's take a clue from phonology: create locality with **tiers**. (Goldsmith 1985; Heinz et al. 2011)
- Enforce constraints on tier, rather than string

Jeff Heinz

Example: Samala Revisited

1. Project sibilant tier
2. *sʃ, *sʒ, *zʃ, *zʒ, *ʃs, *ʒs, *ʃz, *ʒz

*$ h a s x i n t i l a w a ʃ $ $ h a ʃ x i n t i l a w a ʃ $

Making Long-Distance Dependencies Local

- Let's take a clue from phonology: create locality with **tiers**. (Goldsmith 1985; Heinz et al. 2011)
- Enforce constraints on tier, rather than string

Jeff Heinz

Example: Samala Revisited

1. Project sibilant tier
2. *sʃ, *sʒ, *zʃ, *zʒ, *ʃs, *ʒs, *ʃz, *ʒz

*$ h a s x i n t i l a w a ʃ $ $ h a ʃ x i n t i l a w a ʃ $

Making Long-Distance Dependencies Local

- Let's take a clue from phonology: create locality with **tiers**.
  (Goldsmith 1985; Heinz et al. 2011)
- Enforce constraints on tier, rather than string

Jeff Heinz

Example: Samala Revisited

1. Project sibilant tier
2. *sʃ, *sʒ, *zʃ, *zʒ, *ʃs, *ʒs, *ʃz, *ʒz

```
  $     s                 ʃ $
  |     |                 | |
* $ h a s x i n t i l a w a ʃ $        $ h a ʃ x i n t i l a w a ʃ $
```

Making Long-Distance Dependencies Local

- Let's take a clue from phonology: create locality with **tiers**. (Goldsmith 1985; Heinz et al. 2011)
- Enforce constraints on tier, rather than string

Jeff Heinz

Example: Samala Revisited

1. Project sibilant tier
2. *sʃ, *sʒ, *zʃ, *zʒ, *ʃs, *ʒs, *ʃz, *ʒz

```
  $   s          ʃ $        $   ʃ            ʃ $
  |   |          | |        |   |            | |
*$ h a s x i n t i l a w a ʃ $     $ h a ʃ x i n t i l a w a ʃ $
```

Making Long-Distance Dependencies Local

- Let's take a clue from phonology: create locality with **tiers**. (Goldsmith 1985; Heinz et al. 2011)
- Enforce constraints on tier, rather than string

Jeff Heinz

Example: Samala Revisited

1. Project sibilant tier
2. *sʃ, *sʒ, *zʃ, *zʒ, *ʃs, *ʒs, *ʃz, *ʒz

$ s ʃ $ $ ʃ ʃ $

* $ h a s x i n t i l a w a ʃ $ $ h a ʃ x i n t i l a w a ʃ $

Making Long-Distance Dependencies Local

- Let's take a clue from phonology: create locality with **tiers**. (Goldsmith 1985; Heinz et al. 2011)
- Enforce constraints on tier, rather than string

Jeff Heinz

Example: Samala Revisited

1. Project sibilant tier
2. *sʃ, *sʒ, *zʃ, *zʒ, *ʃs, *ʒs, *ʃz, *ʒz

$ s ʃ $

*$ h a s x i n t i l a w a ʃ $

$ ʃ ʃ $

$ h a ʃ x i n t i l a w a ʃ $

Making Long-Distance Dependencies Local

- Let's take a clue from phonology: create locality with **tiers**. (Goldsmith 1985; Heinz et al. 2011)
- Enforce constraints on tier, rather than string

Jeff Heinz

Example: Samala Revisited

1. Project sibilant tier
2. *sʃ, *sʒ, *zʃ, *zʒ, *ʃs, *ʒs, *ʃz, *ʒz

$ s ʃ $ $ ʃ ʃ $

*$ h a s x i n t i l a w a ʃ $ $ h a ʃ x i n t i l a w a ʃ $

Making Long-Distance Dependencies Local

- Let's take a clue from phonology: create locality with **tiers**. (Goldsmith 1985; Heinz et al. 2011)
- Enforce constraints on tier, rather than string

Jeff Heinz

Example: Samala Revisited

1. Project sibilant tier
2. *sʃ, *sʒ, *zʃ, *zʒ, *ʃs, *ʒs, *ʃz, *ʒz

$ s ʃ $

*$ h a s x i n t i l a w a ʃ $

$ ʃ ʃ $

$ h a ʃ x i n t i l a w a ʃ $

Making Long-Distance Dependencies Local

- Let's take a clue from phonology: create locality with **tiers**. (Goldsmith 1985; Heinz et al. 2011)
- Enforce constraints on tier, rather than string

Jeff Heinz

Example: Samala Revisited

1. Project sibilant tier
2. *sʃ, *sʒ, *zʃ, *zʒ, *ʃs, *ʒs, *ʃz, *ʒz

$ s ʃ $ $ ʃ ʃ $

*$ h a s x i n t i l a w a ʃ $ $ h a ʃ x i n t i l a w a ʃ $

What may Project?

Tier projection controlled by

1. label of segment
2. local context
3. symbols already on tier

What may Project?

Tier projection controlled by

1. label of segment
2. local context
3. symbols already on tier

TSL
1

What may Project?

Tier projection controlled by

1. label of segment
2. local context
3. symbols already on tier

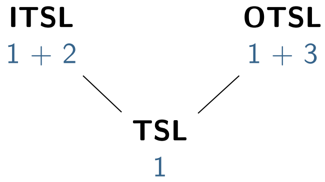
ITSL
$1 + 2$

TSL
1

What may Project?

Tier projection controlled by

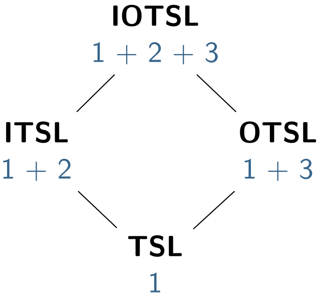
1. label of segment
2. local context
3. symbols already on tier



What may Project?

Tier projection controlled by

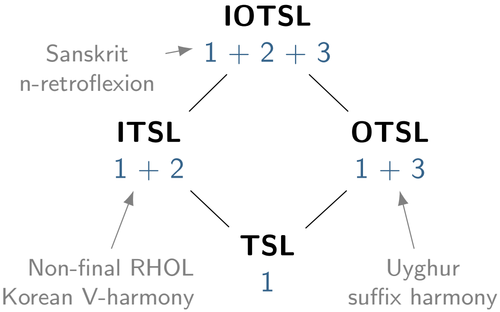
1. label of segment
2. local context
3. symbols already on tier



What may Project?

Tier projection controlled by

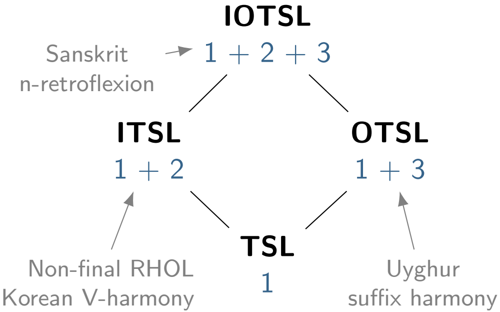
1. label of segment
2. local context
3. symbols already on tier



What may Project?

Tier projection controlled by

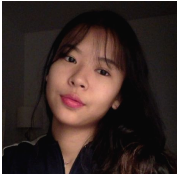
1. label of segment
2. local context
3. symbols already on tier



Hyunah Baek

Aniello De Santo

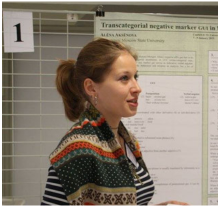
Connor Mayer



Suji Yang

TSL Across Language Modules

- Phonological dependencies fall within the TSL region.
- Morphological dependencies do, too.
  (Aksënova et al. 2016; Aksënova and De Santo 2017; Chandlee 2017)
- Phonology and morphology are **computationally similar**.



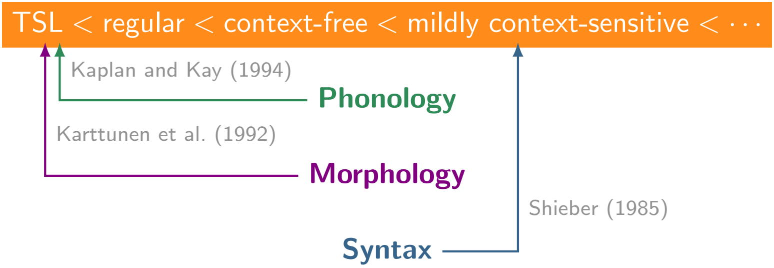
Alëna Aksënova

Sophie Moradi

Why is TSL Relevant?

- Linguistically natural
- Captures wide range of phonotactic dependencies
- Correct and efficient learning algorithms
  (I/O-TSL work in progress)
  (Jardine and McMullin 2017)
- Low resource demand
  - remember the last n symbols of a specific type
  - requires little working memory
  - no complex memory architecture
- Rules out unattested patterns
  - center embedding
  - harmony only if separated by even number of segments

Could Syntax Also be Subregular?



- Syntax seems even more like an outlier. . .
- Don't look at strings!
  What about **syntactic dependencies**?

Nazila Shafiei

c-Strings

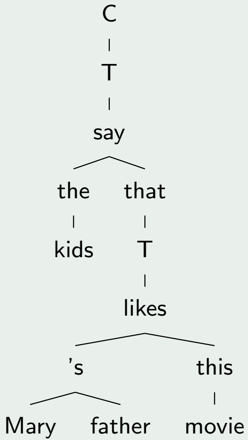
Command-Strings

The **c[ommand]-string** of a node n contains

- n and
- every node that commands n.

- easily computed from dependency trees
- c-command constraints seem to be largely **IOTSL over c-strings**

Example



c-Strings

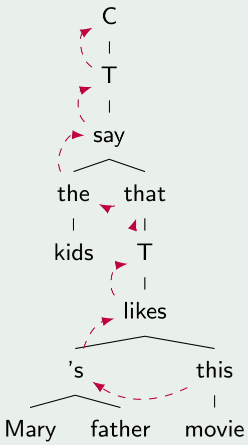
Command-Strings

The **c[ommand]-string** of a node n contains

- n and
- every node that commands n.

- easily computed from dependency trees
- c-command constraints seem to be largely **IOTSL over c-strings**

Example



this 's likes T that the say T C

Principle A

Principle A (as a distributional constraint)

Every reflexive must be c-commanded by a DP in the same TP.

Equivalent c-String Constraint

If the c-string starts with a reflexive,
then at least one D must occur before the first T.

TSL Strategy for Principle A

1. Always project first symbol (ITSL)
2. Project D/T if previous symbol is Refl (OTSL)
3. Constraint: ***Refl T** (bigram)

Principle A

Principle A (as a distributional constraint)

Every reflexive must be c-commanded by a DP in the same TP.

Equivalent c-String Constraint

If the c-string starts with a reflexive,
then at least one D must occur before the first T.

TSL Strategy for Principle A

1. Always project first symbol (ITSL)
2. Project D/T if previous symbol is Refl (OTSL)
3. Constraint: ***Refl T** (bigram)

Principle A

Principle A (as a distributional constraint)

Every reflexive must be c-commanded by a DP in the same TP.

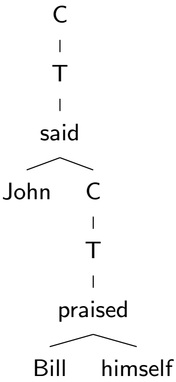
Equivalent c-String Constraint

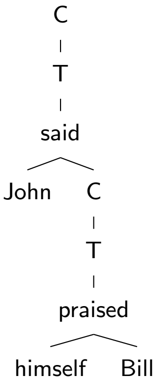
If the c-string starts with a reflexive,
then at least one D must occur before the first T.

TSL Strategy for Principle A

1. Always project first symbol (ITSL)
2. Project D/T if previous symbol is Refl (OTSL)
3. Constraint: *$\textbf{Refl}$ $\textbf{T}$ (bigram)

Example of Principle A as a TSL Constraint





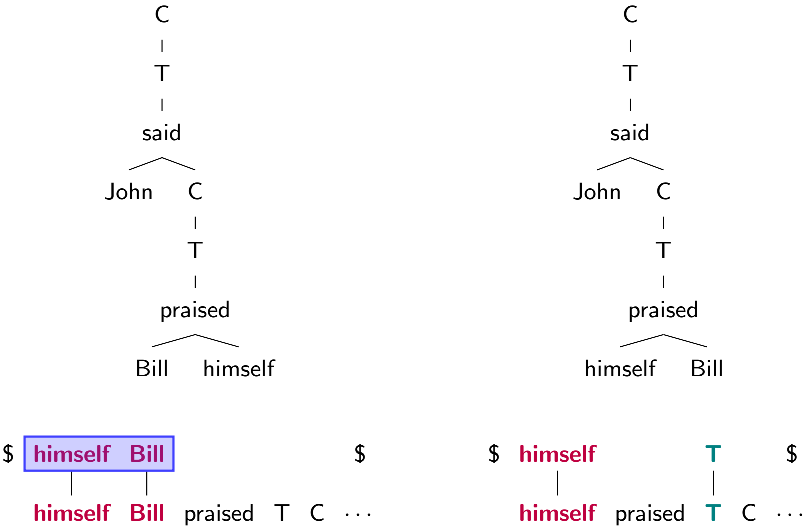
$ **himself** **Bill** $

himself **Bill** praised T C · · ·

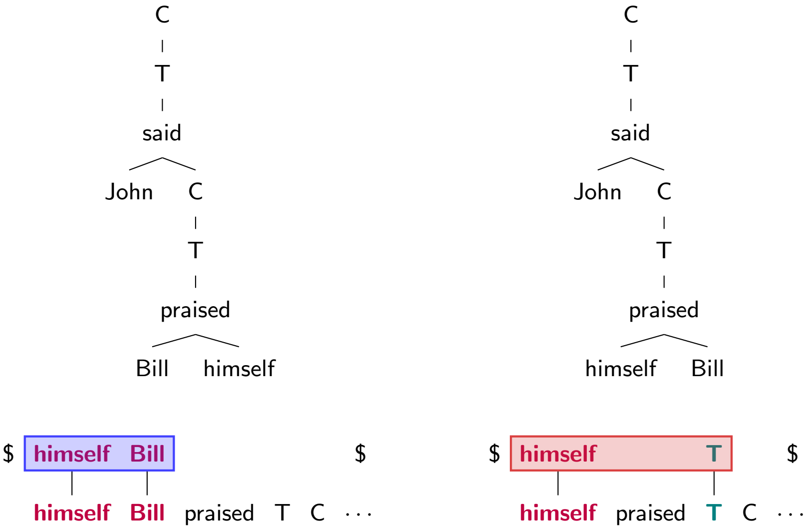
$ **himself** **T** $

himself praised **T** C · · ·

Example of Principle A as a TSL Constraint



Example of Principle A as a TSL Constraint



Another Example: Swedish *sig*

- Swedish *sig* must be non-locally bound.

(1) a. John said Bill praised sig.

b. * Bill praised sig.

TSL Strategy for *sig*

1. Always project first symbol (ITSL)
2. Project T if previous symbol is sig (OTSL)
3. Project D if previous symbol is T (OTSL)
4. Constraint: ***sig T $** (trigram)

$	**sig**			**T**		**John**	$
	sig	Bill	praised	**T**	C	**John**	···

$	**sig**			**T**		**$**
	sig	Bill	praised	**T**	C	

Another Example: Swedish *sig*

- Swedish *sig* must be non-locally bound.

(1) a. John said Bill praised sig.
    b. * Bill praised sig.

TSL Strategy for *sig*

1. Always project first symbol (ITSL)
2. Project T if previous symbol is sig (OTSL)
3. Project D if previous symbol is T (OTSL)
4. Constraint: ***sig T $** (trigram)

$ $\boxed{\textbf{sig} \quad \textbf{T} \quad \textbf{John}}$ $

sig Bill praised **T** C **John** · · ·

$ **sig** **T** **$**

sig Bill praised **T** C

Another Example: Swedish *sig*

- Swedish *sig* must be non-locally bound.

(1) a. John said Bill praised sig.
  b. * Bill praised sig.

TSL Strategy for *sig*

1. Always project first symbol (ITSL)
2. Project T if previous symbol is sig (OTSL)
3. Project D if previous symbol is T (OTSL)
4. Constraint: ***sig T $** (trigram)

$ **sig** **T** **John** $
sig Bill praised **T** C **John** ...

$ **sig** **T** **$**
sig Bill praised **T** C

Comparison to Phonology and Morphology

Similarities

- mostly bigram and trigram constraints
- simple structural contexts
- dependencies in phonology are also c-command-like (Graf 2018a)

Differences

- OTSL seems more common in syntax

A Typological Prediction

Formal typology of syntactic constraints
should mirror phonology and morphology.

Connection to Parsing

- A tree is well-formed only if
  each node has a well-formed c-string.
- verifiable by deterministic top-down tree automaton
  with finite look-ahead
  $\Rightarrow$ **efficient incremental parsing**

An Intriguing Hypothesis

- Why c-command (rather than, say, precedence)?
- Because it allows for more efficient processing!

- But syntax isn't just c-command.
  There's also displacement/movement...

Connection to Parsing

- A tree is well-formed only if each node has a well-formed c-string.
- verifiable by deterministic top-down tree automaton with finite look-ahead
  $\Rightarrow$ **efficient incremental parsing**

An Intriguing Hypothesis

- Why c-command (rather than, say, precedence)?
- Because it allows for more efficient processing!

- But syntax isn't just c-command.
  There's also displacement/movement...

Connection to Parsing

- A tree is well-formed only if
  each node has a well-formed c-string.
- verifiable by deterministic top-down tree automaton
  with finite look-ahead
  $\Rightarrow$ **efficient incremental parsing**

An Intriguing Hypothesis

- Why c-command (rather than, say, precedence)?
- Because it allows for more efficient processing!

- But syntax isn't just c-command.
  There's also displacement/movement...

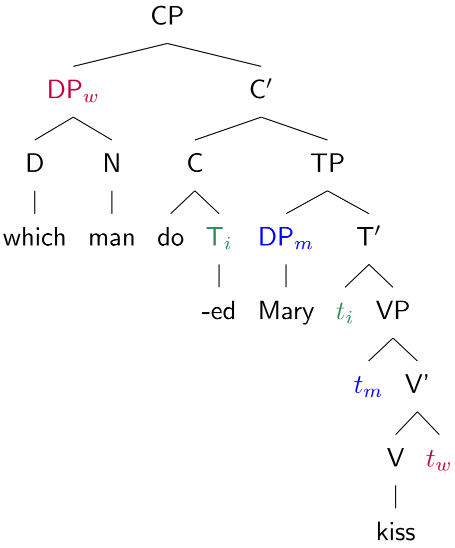
Minimalist Grammars

Ed Stabler

- Minimalist grammars (MGs) are a formalization of Minimalist syntax. (Stabler 1997, 2011)
- Operations: **Merge** and **Move**
- Adopt Chomsky-Borer hypothesis: Grammar is just a finite list of feature-annotated lexical items

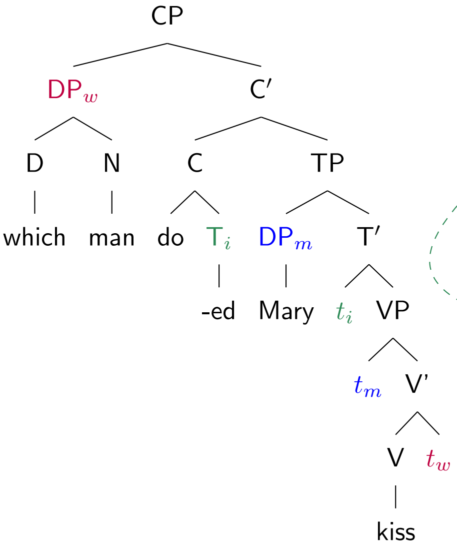
Chemistry	Syntax
atoms	words
electrons	features
molecules	sentences

MG Syntax in Action

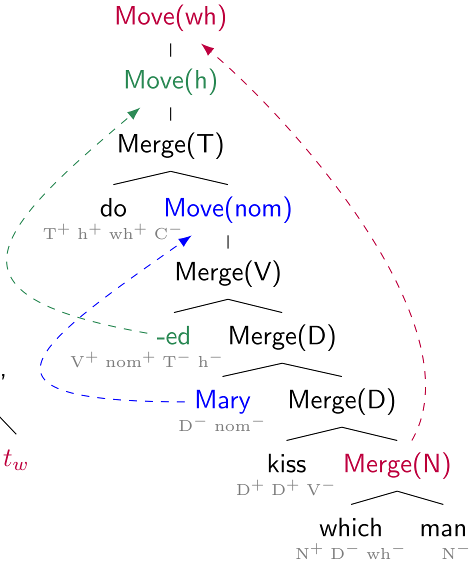


Phrase Structure Tree

MG Syntax in Action



Phrase Structure Tree

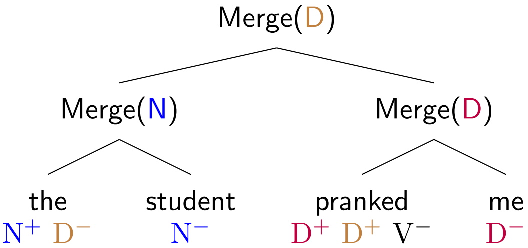


Derivation Tree

The Central Role of Derivation Trees

- Derivation trees are rarely considered in generative syntax.
  (but see Epstein et al. 1998)
- Satisfy Chomsky's structural desiderata:
  - no linear order
  - label-free
  - extension condition
  - inclusiveness condition
- Contain all information to produce phrase structure trees
  $\Rightarrow$ **central data structure** of Minimalist syntax

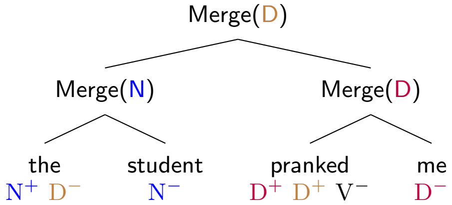
Merge is TSL



- The selector features of the head have to match the category features of the arguments.
- 1-to-1 match between selector features and category features.
- This is naturally expressed as **TSL over trees**.

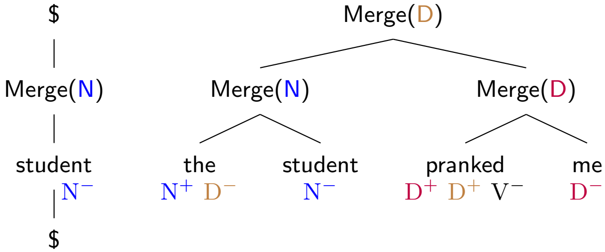
Category Tiers for Merge

- Project tree tier for each category X.
- Every X^- has a Merge node as its mother.
- Every Merge node has exactly one X^- among its daughters.



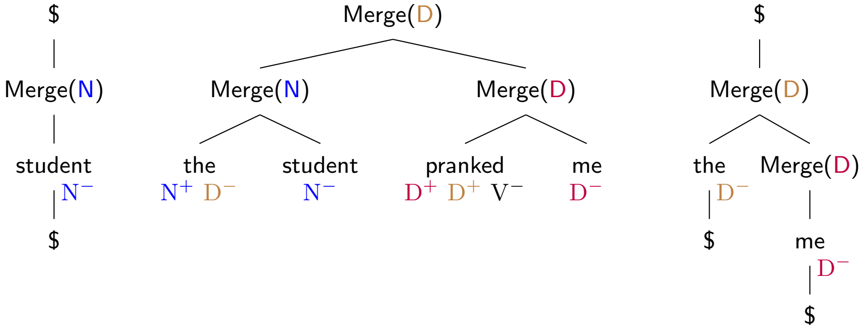
Category Tiers for Merge

- Project tree tier for each category X.
- Every X^- has a Merge node as its mother.
- Every Merge node has exactly one X^- among its daughters.

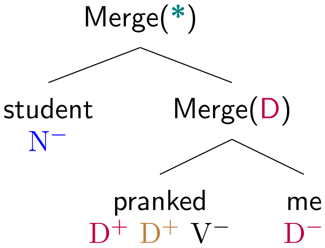


Category Tiers for Merge

- Project tree tier for each category X.
- Every X^- has a Merge node as its mother.
- Every Merge node has exactly one X^- among its daughters.

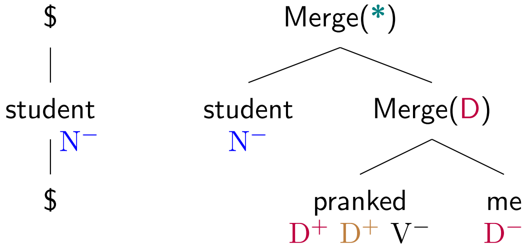


Illicit Merge Yields Ill-Formed Tiers



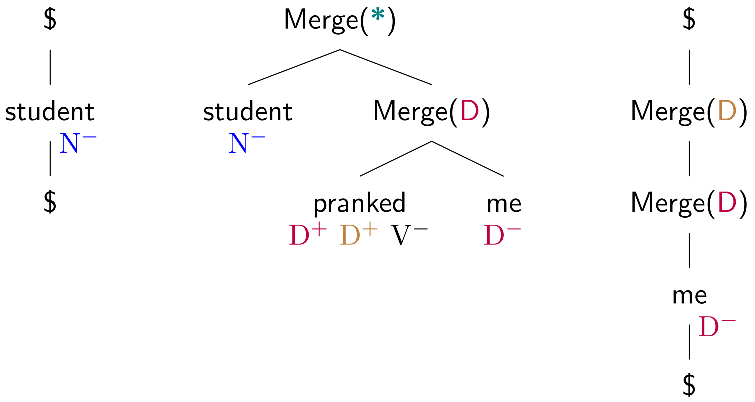
- This handles Merge.
- Moving on to Move. . .

Illicit Merge Yields Ill-Formed Tiers



- This handles Merge.
- Moving on to Move. . .

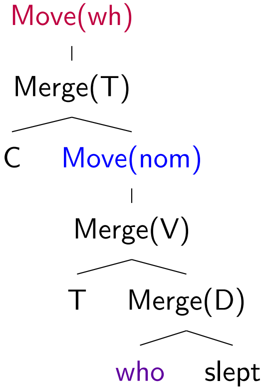
Illicit Merge Yields Ill-Formed Tiers



- This handles Merge.
- Moving on to Move. . .

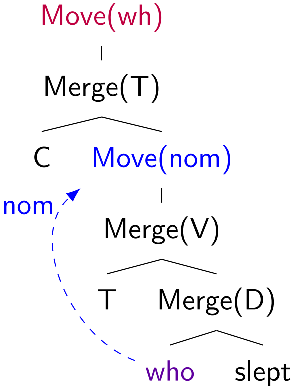
Move: Single Movement Normal Form

- **Assumption:** every phrase at most one movement feature
- Intermediate landing sites not feature-triggered
  (Graf et al. 2016)



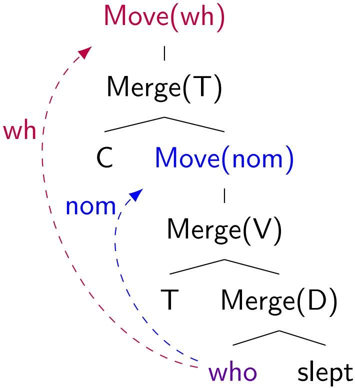
Move: Single Movement Normal Form

- **Assumption:** every phrase at most one movement feature
- Intermediate landing sites not feature-triggered
  (Graf et al. 2016)



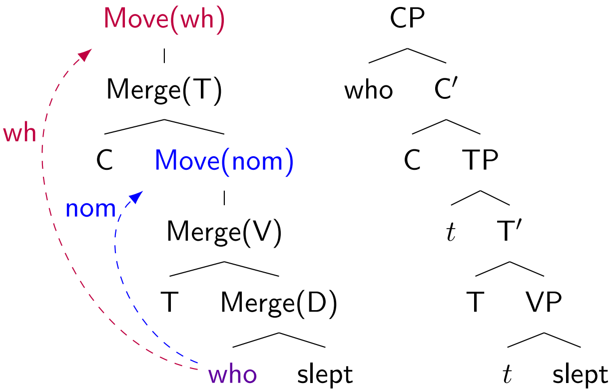
Move: Single Movement Normal Form

- **Assumption:** every phrase at most one movement feature
- Intermediate landing sites not feature-triggered
  (Graf et al. 2016)



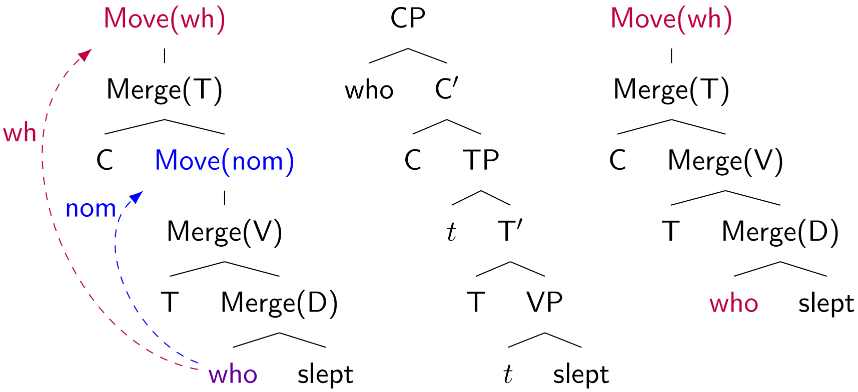
Move: Single Movement Normal Form

- **Assumption:** every phrase at most one movement feature
- Intermediate landing sites not feature-triggered
  (Graf et al. 2016)



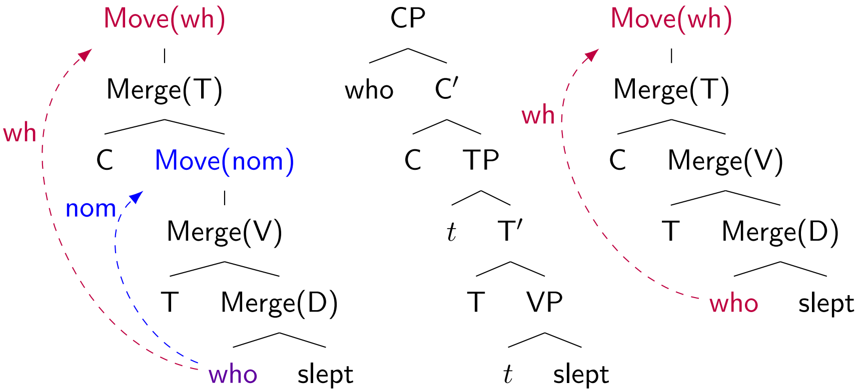
Move: Single Movement Normal Form

- **Assumption:** every phrase at most one movement feature
- Intermediate landing sites not feature-triggered
  (Graf et al. 2016)



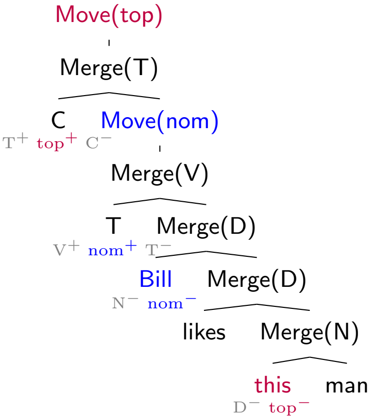
Move: Single Movement Normal Form

- **Assumption:** every phrase at most one movement feature
- Intermediate landing sites not feature-triggered (Graf et al. 2016)



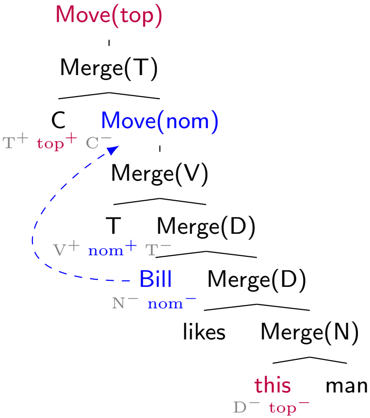
Movement Tiers for Move

- Project tree tier for each movement type x.
- Every x^- has a Move node as its mother.
- Every Move node has exactly one x^- among its daughters.



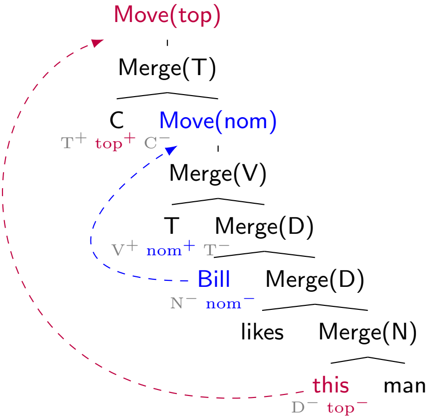
Movement Tiers for Move

- Project tree tier for each movement type x.
- Every x^- has a Move node as its mother.
- Every Move node has exactly one x^- among its daughters.



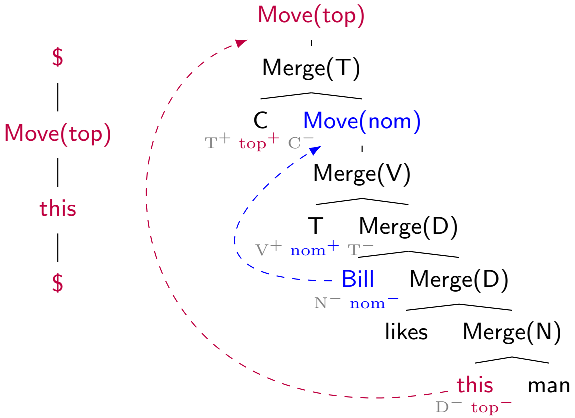
Movement Tiers for Move

- Project tree tier for each movement type x.
- Every x^- has a Move node as its mother.
- Every Move node has exactly one x^- among its daughters.



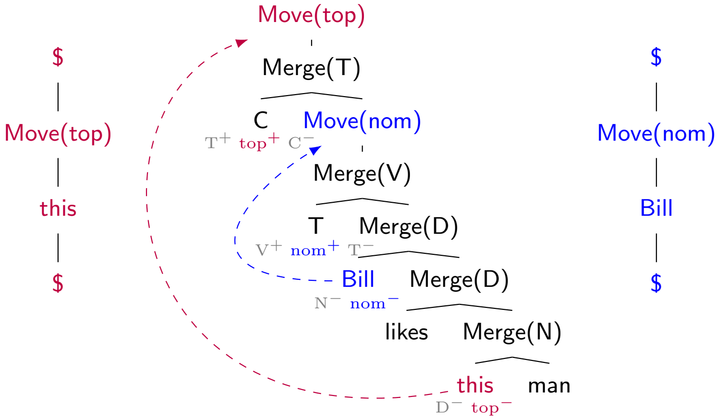
Movement Tiers for Move

- Project tree tier for each movement type x.
- Every x^- has a Move node as its mother.
- Every Move node has exactly one x^- among its daughters.

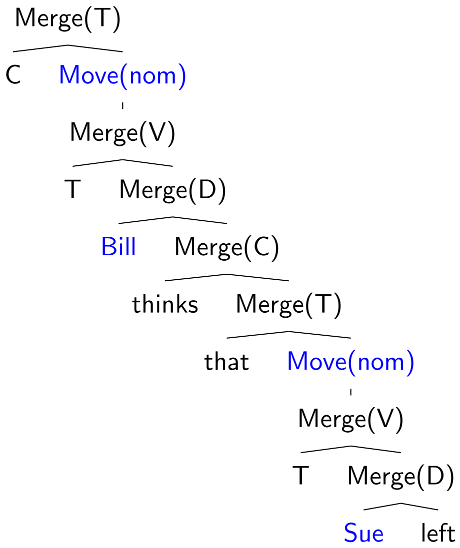


Movement Tiers for Move

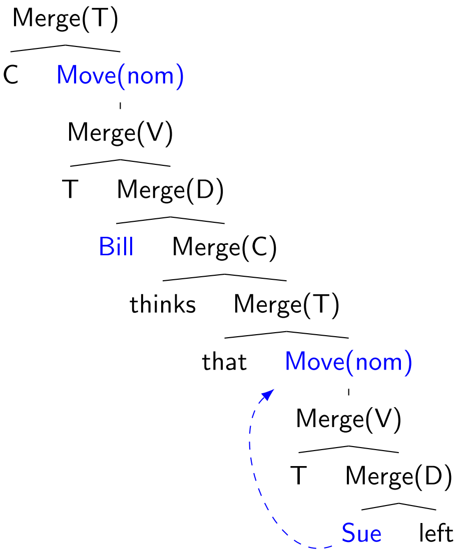
- Project tree tier for each movement type x.
- Every x^- has a Move node as its mother.
- Every Move node has exactly one x^- among its daughters.



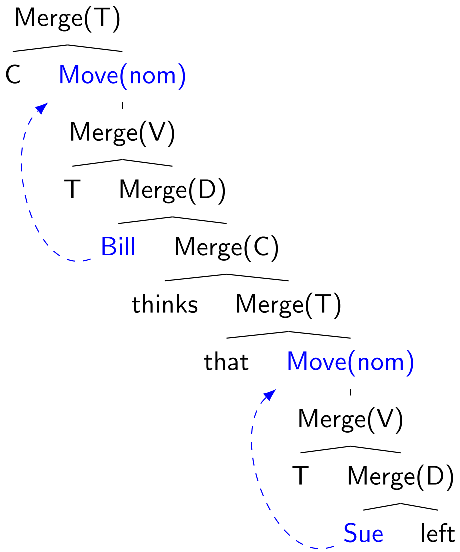
A Tier With Multiple Movers



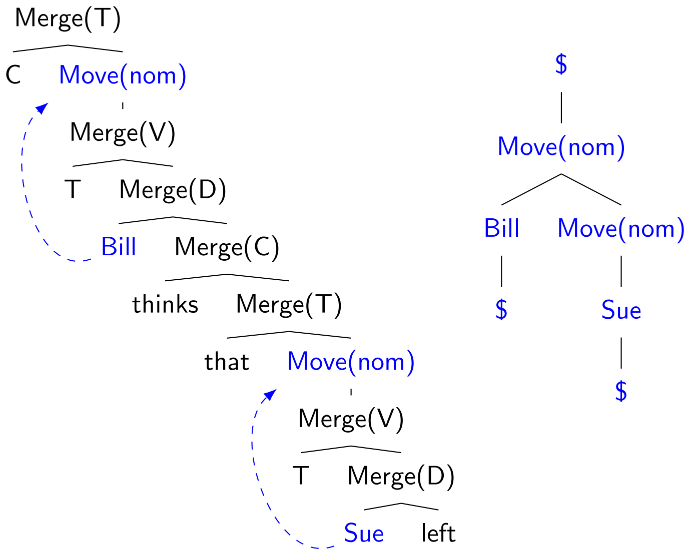
A Tier With Multiple Movers



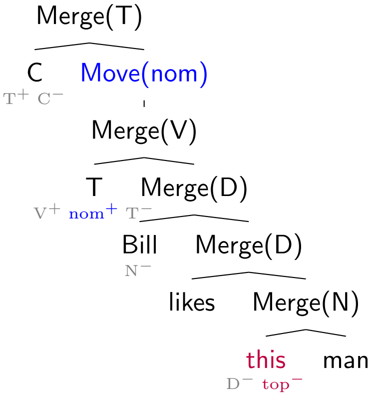
A Tier With Multiple Movers



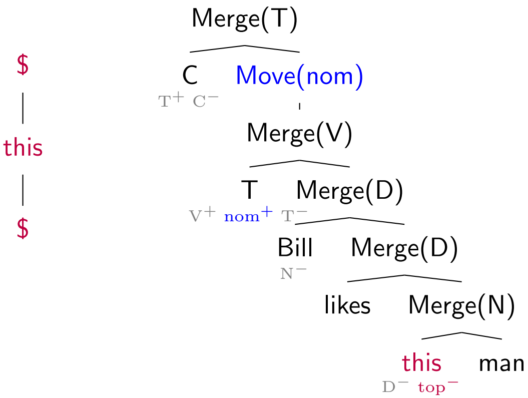
A Tier With Multiple Movers



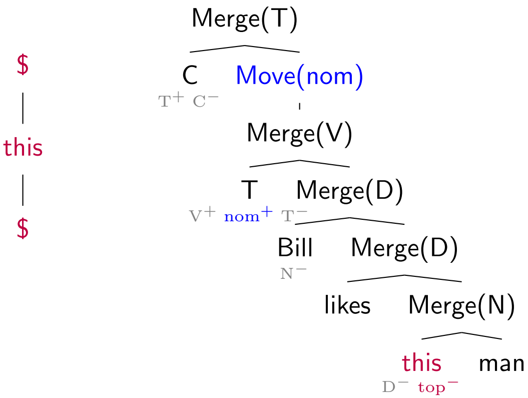
Blocking Simple Cases of Illicit Movement



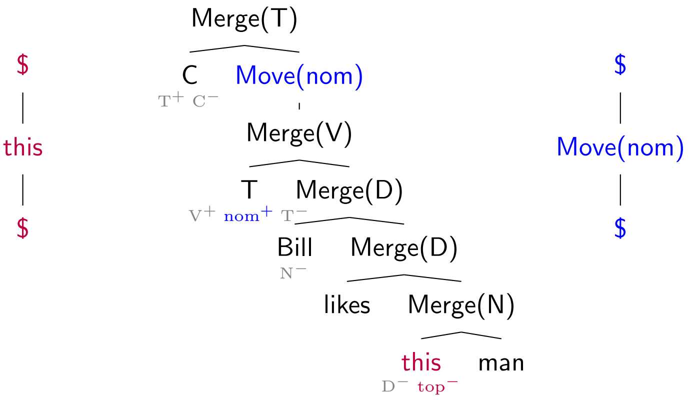
Blocking Simple Cases of Illicit Movement



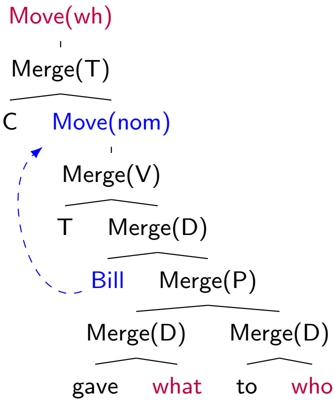
Blocking Simple Cases of Illicit Movement



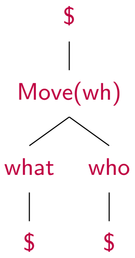
Blocking Simple Cases of Illicit Movement

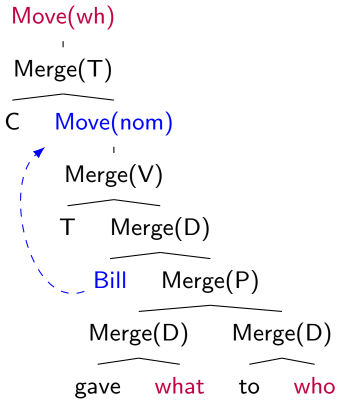


Blocking Many-to-One Movement

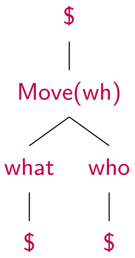


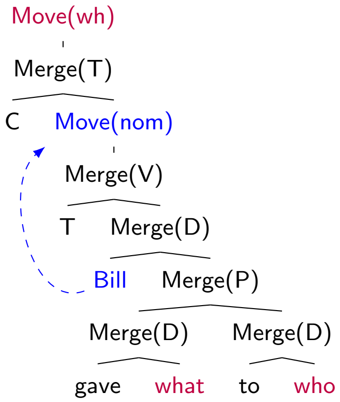
Blocking Many-to-One Movement



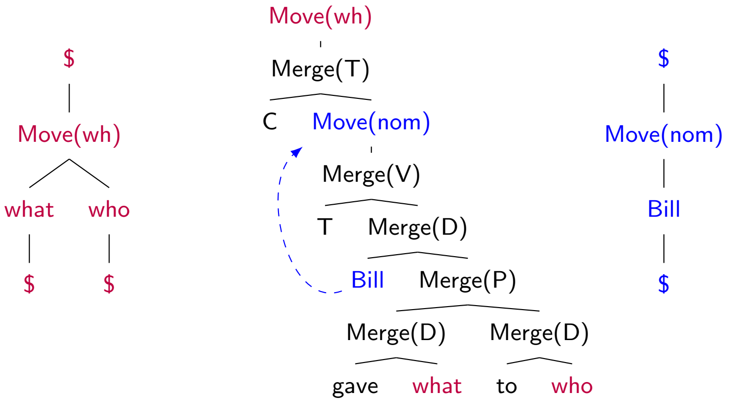


Blocking Many-to-One Movement





Blocking Many-to-One Movement



The Common Core of Merge and Move

TSL Strategy for Merge

- Project tree tier for each **category** X.
- Every X^- has a **Merge** node as its mother.
- Every **Merge** node has exactly one X^- among its daughters.

TSL Strategy for Move

- Project tree tier for each **movement type** x.
- Every x^- has a **Move** node as its mother.
- Every **Move** node has exactly one x^- among its daughters.

Note: constraints again **highly local**

Summary

- Syntax looks like a complex outlier.
- But not if we choose appropriate representations:
  - c-command dependencies are TSL over c-strings
  - Merge and Move are TSL over derivation trees
- **Computational parallelism**:
  - phonology is TSL
  - morphology is TSL
  - syntax is TSL

Work In Progress

- **Movement**
  - Interaction of movement and c-command
  - Complexity without Single Movement Normal Form
- **Empirical work**
  - limits of c-string constraints
  - unified treatment of island constraints
  - modeling specific phenomena (e.g. case assignment)
- **Processing & Learning**
  - compiling c-string constraints into MG parser
  - learning via semantic bootstrapping

Open Issues

- experimental evidence for computational parallelism
- even tighter subclasses of TSL
- full predicted typology
- model concrete aspects of acquisition

Open Issues

- experimental evidence for computational parallelism
- even tighter subclasses of TSL
- full predicted typology
- model concrete aspects of acquisition

Join the program!

Resources and Readings

1. **Survey papers:** Pullum and Rogers (2006); Heinz (2011a,b, 2018); Rogers and Pullum (2011); Chandlee and Heinz (2016)
2. **TSL and its extensions:** Heinz et al. (2011); McMullin (2016); Baek (2017); De Santo (2017); De Santo and Graf (2017); Graf (2017c); Graf and Mayer (2018); Mayer and Major (2018); Yang (2018)
3. **TSL morphology:** Aksënova et al. (2016); Graf (2017b)
4. **TSL morpho-semantics:** Graf (2017d)
5. **TSL syntax:** Graf (2012a, 2018b); Graf and Shafiei (2019); Vu (2018); Vu et al. (2019)
6. **Mappings:** Courcelle and Engelfriet (2012); Chandlee (2014, 2017); Jardine (2016)
7. **Learnability:** Heinz (2010); Kasprzik and Kötzing (2010); Heinz et al. (2012); Jardine et al. (2014); Lai (2015); Jardine and Heinz (2016); Jardine and McMullin (2017)

Appendix

Psychological Reality of Derivation Trees

Central role of derivation trees backed up by **processing data**:

- Derivation trees can be parsed top-down (Stabler 2013)
- Parsing models update Derivational Theory of Complexity, make correct processing predictions for
  - right $<$ center embedding (Kobele et al. 2013)
  - crossing $<$ nested dependencies (Kobele et al. 2013)
  - SC-RC $<$ RC-SC (Graf et al. 2017)
  - SRC $<$ ORC in English (Graf et al. 2017)
  - SRC $<$ ORC in East-Asian (Graf et al. 2017)
  - quantifier scope preferences (Pasternak 2016)
  - stacked relative clauses (Zhang 2017)
  - Korean attachment ambiguities

Technical Fertility of Derivation Trees

Derivation trees made it easy for MGs to accommodate the full syntactic toolbox:

- sidewards movement (Stabler 2006; Graf 2013)
- affix hopping (Graf 2012b, 2013)
- clustering movement (Gärtner and Michaelis 2010)
- tucking in (Graf 2013)
- ATB movement (Kobele 2008)
- copy movement (Kobele 2006)
- extraposition (Hunter and Frank 2014)
- Late Merge (Kobele 2010; Graf 2014a)
- Agree (Kobele 2011; Graf 2012a)
- adjunction (Fowlie 2013; Graf 2014b; Hunter 2015)
- TAG-style adjunction (Graf 2012c)

Even More MG Extensions

- local and global constraints (Kobele 2011; Graf 2012a, 2017a)
- transderivational constraints (Graf 2010, 2013)
- Principle A and B (Graf and Abner 2012)
- GPSG-style feature percolation (Kobele 2008)
- idioms (Kobele 2012)
- grafts (multi-rooted multi-dominance trees) (Graf in progress)

Long Story Short

Derivation trees are a more useful and fertile data structure than phrase structure trees.

Even More MG Extensions

- local and global constraints (Kobele 2011; Graf 2012a, 2017a)
- transderivational constraints (Graf 2010, 2013)
- Principle A and B (Graf and Abner 2012)
- GPSG-style feature percolation (Kobele 2008)
- idioms (Kobele 2012)
- grafts (multi-rooted multi-dominance trees) (Graf in progress)

Long Story Short

Derivation trees are a more useful and fertile data structure than phrase structure trees.

References I

Aksënova, Alëna, and Aniello De Santo. 2017. Strict locality in morphological derivations. URL https://linguistics.stonybrook.edu/sites/default/files/uploads/u105/cls-SL-pres.pdf, slides of a talk presented at CLS 2017.

Aksënova, Alëna, Thomas Graf, and Sedigheh Moradi. 2016. Morphotactics as tier-based strictly local dependencies. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, 121–130. URL https://www.aclweb.org/anthology/W/W16/W16-2019.pdf.

Baek, Hyunah. 2017. Computational representation of unbounded stress: Tiers with structural features. Ms., Stony Brook University; to appear in *Proceedings of CLS 53*.

Chandlee, Jane. 2014. *Strictly local phonological processes*. Doctoral Dissertation, University of Delaware. URL http://udspace.udel.edu/handle/19716/13374.

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

Chandlee, Jane, and Jeffrey Heinz. 2016. Computational phonology. Ms., Haverford College and University of Delaware.

Courcelle, Bruno, and Joost Engelfriet. 2012. *Graph structure and monadic second-order logic: A language-theoretic approach*. Cambridge, UK: Cambridge University Press.

References II

De Santo, Aniello. 2017. Extending TSL languages: Conjunction as multiple tier-projection. Ms., Stony Brook University.

De Santo, Aniello, and Thomas Graf. 2017. Structure sensitive tier projection: Applications and formal properties. Ms., Stony Brook University.

Epstein, Samuel D., Erich M. Groat, Ruriko Kawashima, and Hisatsugu Kitahara. 1998. *A derivational approach to syntactic relations*. Oxford: Oxford University Press.

Fowlie, Meaghan. 2013. Order and optionality: Minimalist grammars with adjunction. In *Proceedings of the 13th Meeting on the Mathematics of Language (MoL 13)*, ed. András Kornai and Marco Kuhlmann, 12–20.

Gärtner, Hans-Martin, and Jens Michaelis. 2010. On the treatment of multiple-wh-interrogatives in Minimalist grammars. In *Language and logos*, ed. Thomas Hanneforth and Gisbert Fanselow, 339–366. Berlin: Akademie Verlag.

Goldsmith, John. 1985. A principled exception to the coordinate structure constraint. In *Papers from the Twenty-First Annual Regional Meeting of the Chicago Linguistic Society*, 133–143.

Graf, Thomas. 2010. A tree transducer model of reference-set computation. *UCLA Working Papers in Linguistics* 15:1–53.

References III

Graf, Thomas. 2012a. Locality and the complexity of Minimalist derivation tree languages. In *Formal Grammar 2010/2011*, ed. Philippe de Groot and Mark-Jan Nederhof, volume 7395 of *Lecture Notes in Computer Science*, 208–227. Heidelberg: Springer. URL `http://dx.doi.org/10.1007/978-3-642-32024-8_14`.

Graf, Thomas. 2012b. Movement-generalized Minimalist grammars. In *LACL 2012*, ed. Denis Béchet and Alexander J. Dikovsky, volume 7351 of *Lecture Notes in Computer Science*, 58–73. URL `http://dx.doi.org/10.1007/978-3-642-31262-5_4`.

Graf, Thomas. 2012c. Tree adjunction as Minimalist lowering. In *Proceedings of the 11th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+11)*, 19–27. URL `http://www.aclweb.org/old_anthology/W/W12/W12-4603.pdf`.

Graf, Thomas. 2013. *Local and transderivational constraints in syntax and semantics*. Doctoral Dissertation, UCLA. URL `http://thomasgraf.net/doc/papers/Graf13Thesis.pdf`.

Graf, Thomas. 2014a. Late merge as lowering movement in Minimalist grammars. In *LACL 2014*, ed. Nicholas Asher and Sergei Soloviev, volume 8535 of *Lecture Notes in Computer Science*, 107–121. Heidelberg: Springer. URL `https://doi.org/10.1007/978-3-662-43742-1_9`.

References IV

Graf, Thomas. 2014b. Models of adjunction in Minimalist grammars. In *Formal Grammar 2014*, ed. Glynn Morrill, Reinhard Muskens, Rainer Osswald, and Frank Richter, volume 8612 of *Lecture Notes in Computer Science*, 52–68. Heidelberg: Springer. URL `https://doi.org/10.1007/978-3-662-44121-3_4`.

Graf, Thomas. 2017a. A computational guide to the dichotomy of features and constraints. *Glossa* 2:1–36. URL `https://dx.doi.org/10.5334/gjgl.212`.

Graf, Thomas. 2017b. Graph transductions and typological gaps in morphological paradigms. In *Proceedings of the 15th Meeting on the Mathematics of Language (MOL 2017)*, 114–126. URL `http://www.aclweb.org/anthology/W17-3411`.

Graf, Thomas. 2017c. The power of locality domains in phonology. *Phonology* 34:385–405. URL `https://dx.doi.org/10.1017/S0952675717000197`.

Graf, Thomas. 2017d. The subregular complexity of monomorphemic quantifiers. URL `http://thomasgraf.net/doc/papers/Graf17SP.pdf`, ms., Stony Brook University.

Graf, Thomas. 2018a. Locality domains and phonological c-command over strings. In *Proceedings of NELS 2017*. URL `http://ling.auf.net/lingbuzz/004080`, to appear.

Graf, Thomas. 2018b. Why movement comes for free once you have adjunction. In *Proceedings of CLS 53*. URL `http://ling.auf.net/lingbuzz/003943`, to appear.

References V

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. URL http://thomasgraf.net/doc/papers/GrafAbner12TAG.pdf.

Graf, Thomas, Alëna Aksënova, and Aniello De Santo. 2016. A single movement normal form for Minimalist grammars. In *Formal Grammar: 20th and 21st International Conferences, FG 2015, Barcelona, Spain, August 2015, Revised Selected Papers. FG 2016, Bozen, Italy, August 2016*, ed. Annie Foret, Glyn Morrill, Reinhard Muskens, Rainer Osswald, and Sylvain Pogodalla, 200–215. Berlin, Heidelberg: Springer. URL https://doi.org/10.1007/978-3-662-53042-9_12.

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

Graf, Thomas, James Monette, and Chong Zhang. 2017. Relative clauses as a benchmark for Minimalist parsing. *Journal of Language Modelling* 5:57–106. URL http://dx.doi.org/10.15398/jlm.v5i1.157.

Graf, Thomas, and Nazila Shafiei. 2019. C-command dependencies as TSL string constraints. In *Proccedings of SCiL 2019*. To appear.

Heinz, Jeffrey. 2010. String extension learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 897–906. URL http://www.aclweb.org/anthology/P10-1092.pdf.

References VI

Heinz, Jeffrey. 2011a. Computational phonology — part I: Foundations. *Language and Linguistics Compass* 5:140–152.

Heinz, Jeffrey. 2011b. Computational phonology — part II: Grammars, learning, and the future. *Language and Linguistics Compass* 5:153–168.

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

Heinz, Jeffrey, Anna Kasprzik, and Timo Kötzing. 2012. Learning in the limit with lattice-structured hypothesis spaces. *Theoretical Computer Science* 457:111–127. URL `https://doi.org/10.1016/j.tcs.2012.07.017`.

Heinz, Jeffrey, Chetan Rawal, and Herbert G. Tanner. 2011. Tier-based strictly local constraints in phonology. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, 58–64. URL `http://www.aclweb.org/anthology/P11-2011`.

Hunter, Tim. 2015. Deconstructing merge and move to make room for adjunction. *Syntax* 18:266–319.

Hunter, Tim, and Robert Frank. 2014. Eliminating rightward movement: Extraposition as flexible linearization of adjuncts. *Linguistic Inquiry* 45:227–267.

Jardine, Adam. 2016. Computationally, tone is different. *Phonology* 33:247–283. URL `https://doi.org/10.1017/S0952675716000129`.

References VII

Jardine, Adam, Jane Chandlee, Rémi Eryaud, and Jeffrey Heinz. 2014. Very efficient learning of structured classes of subsequential functions from positive data. In *Proceedings of the 12th International Conference on Grammatical Inference (ICGI 2014)*, JMLR Workshop Proceedings, 94–108. URL http://www.jmlr.org/proceedings/papers/v34/jardine14a.html.

Jardine, Adam, and Jeffrey Heinz. 2016. Learning tier-based strictly 2-local languages. *Transactions of the ACL* 4:87–98. URL https://aclweb.org/anthology/Q/Q16/Q16-1007.pdf.

Jardine, Adam, and Kevin McMullin. 2017. Efficient learning of tier-based strictly k-local languages. In *Proceedings of Language and Automata Theory and Applications*, Lecture Notes in Computer Science, 64–76. Berlin: Springer. URL https://doi.org/10.1007/978-3-319-53733-7_4.

Joshi, Aravind. 1985. Tree-adjoining grammars: How much context sensitivity is required to provide reasonable structural descriptions? In *Natural language parsing*, ed. David Dowty, Lauri Karttunen, and Arnold Zwicky, 206–250. Cambridge: Cambridge University Press.

Kaplan, Ronald M., and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics* 20:331–378. URL http://www.aclweb.org/anthology/J94-3001.pdf.

References VIII

Karttunen, Lauri, Ronald M. Kaplan, and Annie Zaenen. 1992. Two-level morphology with composition. In *COLING'92*, 141–148. URL http://www.aclweb.org/anthology/C92-1025.

Kasprzik, Anna, and Timo Kötzing. 2010. String extension learning using lattices. In *Language and automata theory and applications: 4th international conference, LATA 2010, Trier, Germany, May 24-28, 2010*, ed. Adrian-Horia Dediu, Henning Fernau, and Carlos Martín-Vide, 380–391. Berlin, Heidelberg: Springer. URL http://dx.doi.org/10.1007/978-3-642-13089-2_32.

Kobele, Gregory M. 2006. *Generating copies: An investigation into structural identity in language and grammar*. Doctoral Dissertation, UCLA. URL http://home.uchicago.edu/~gkobele/files/Kobele06GeneratingCopies.pdf.

Kobele, Gregory M. 2008. Across-the-board extraction and Minimalist grammars. In *Proceedings of the Ninth International Workshop on Tree Adjoining Grammars and Related Frameworks*.

Kobele, Gregory M. 2010. On late adjunction in Minimalist grammars. URL http://research.nii.ac.jp/~kanazawa/mcfgplus/2010/2010-Kobele10LateAdjunction.pdf, slides for a talk given at MCFG+ 2010.

Kobele, Gregory M. 2011. Minimalist tree languages are closed under intersection with recognizable tree languages. In *LACL 2011*, ed. Sylvain Pogodalla and Jean-Philippe Prost, volume 6736 of *Lecture Notes in Artificial Intelligence*, 129–144. URL https://doi.org/10.1007/978-3-642-22221-4_9.

References IX

Kobele, Gregory M. 2012. Idioms and extended transducers. In *Proceedings of the 11th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+11)*, 153–161. Paris, France. URL http://www.aclweb.org/anthology-new/W/W12/W12-4618.

Kobele, Gregory M., Sabrina Gerth, and John T. Hale. 2013. Memory resource allocation in top-down Minimalist parsing. In *Formal Grammar: 17th and 18th International Conferences, FG 2012, Opole, Poland, August 2012, Revised Selected Papers, FG 2013, Düsseldorf, Germany, August 2013*, ed. Glyn Morrill and Mark-Jan Nederhof, 32–51. Berlin, Heidelberg: Springer. URL https://doi.org/10.1007/978-3-642-39998-5_3.

Lai, Regine. 2015. Learnable vs. unlearnable harmony patterns. *Linguistic Inquiry* 46:425–451.

Mayer, Connor, and Travis Major. 2018. A challenge for tier-based strict locality from Uyghur backness harmony. In *Proceedings of Formal Grammar 2018*. To appear.

McMullin, Kevin. 2016. *Tier-based locality in long-distance phonotactics: Learnability and typology*. Doctoral Dissertation, University of British Columbia.

Pasternak, Robert. 2016. Memory usage and scope ambiguity resolution. Qualifying paper, Stony Brook University.

References X

Pullum, Geoffrey K., and James Rogers. 2006. Animal pattern-learning experiments: Some mathematical background. Ms., Radcliffe Institute for Advanced Study, Harvard University.

Rogers, James, and Geoffrey K. Pullum. 2011. Aural pattern recognition experiments and the subregular hierarchy. *Journal of Logic, Language and Information* 20:329–342.

Shieber, Stuart M. 1985. Evidence against the context-freeness of natural language. *Linguistics and Philosophy* 8:333–345. URL `http://dx.doi.org/10.1007/BF00630917`.

Stabler, Edward P. 1997. Derivational Minimalism. In *Logical aspects of computational linguistics*, ed. Christian Retoré, volume 1328 of *Lecture Notes in Computer Science*, 68–95. Berlin: Springer. URL `https://doi.org/10.1007/BFb0052152`.

Stabler, Edward P. 2006. Sidewards without copying. In *Formal Grammar '06, Proceedings of the Conference*, ed. Gerald Penn, Giorgio Satta, and Shuly Wintner, 133–146. Stanford: CSLI.

Stabler, Edward P. 2011. Computational perspectives on Minimalism. In *Oxford handbook of linguistic Minimalism*, ed. Cedric Boeckx, 617–643. Oxford: Oxford University Press.

References XI

Stabler, Edward P. 2013. Two models of minimalist, incremental syntactic analysis. *Topics in Cognitive Science* 5:611–633. URL `https://dx.doi.org/10.1111/tops.12031`.

Vu, Mai Ha. 2018. Towards a formal description of NPI-licensing patterns. In *Proceedings of the Society for Computation in Linguistics*, volume 1, 154–163.

Vu, Mai Ha, Nazila Shafiei, and Thomas Graf. 2019. Case assignment in TSL syntax: A case study. In *Proccedings of SCiL 2019*. To appear.

Yang, Su Ji. 2018. Subregular complexity in Korean phonotactics. Undergraduate honors thesis, Stony Brook University.

Zhang, Chong. 2017. *Stacked relatives: Their structure, processing, and computation*. Doctoral Dissertation, Stony Brook University.