Subregular Linguistics for Linguists

(100% Formula-Free)

Thomas Graf

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

Invited Talk UMass
November 8, 2019

You can get the slides here under “News”
Definition (Synchronous ISL transduction)

A node \( b \) in tree \( u \) can be **targeted** by an ISL-\( k \) context \( \langle s, a, t \rangle \) iff there is some \( p \in \mathbb{N}^* \) such that

- **node match** \( b = pa \), and
- **label match** for all nodes \( g \) of \( s \), \( \ell_s(g) = \ell_u(pg) \),
- **full-width match** for all nodes \( g_i \) of \( s \) with \( g \in \mathbb{N}^* \) and \( i \in \mathbb{N} \), if \( pgj \) is a node of \( u \) \( (j > i) \), then \( gj \) is a node of \( s \).

Now suppose furthermore that \( n \) in \( u \) has \( d \geq 0 \) daughters. Given an ISL-\( k \) tree transducer \( \tau \), we use \( \leftarrow \mathcal{T}(u, b) \) to denote the set of all trees \( t[\square_1 \leftarrow \mathcal{T}(u, b1), \ldots, \square_d \leftarrow \mathcal{T}(u, bd)] \) such that there is a rewrite rule \( \langle s, a, t \rangle \) in \( \tau \) that targets node \( b \) in \( u \). If this set is empty, \( \leftarrow \mathcal{T}(u, b) \) is undefined. For any \( \Sigma \)-tree \( t \), we may simply write \( \leftarrow \mathcal{T}(t) \) instead of \( \leftarrow \mathcal{T}(t, \varepsilon) \).

For any tree language \( L \), the transduction computed by \( \tau \) in **synchronous mode** is \( \leftarrow \mathcal{T}(L) := \{ \langle i, o \rangle \mid i \in L, o \in \leftarrow \mathcal{T}(i) \} \). A transduction is **synchronous input strictly \( k \)-local** (sISL-\( k \)) iff it can be computed by some ISL-\( k \) transducer in synchronous mode.
Definition (Synchronous ISL transduction)

A node $b$ in tree $u$ can be **targeted** by an ISL-$k$ context $\langle s, a, t \rangle$ iff there is some $p \in \mathbb{N}^*$ such that

- **node match** $b = pa$, and
- **label match** for all nodes $g$ of $s$, $\ell_s(g) = \ell_u(pg)$,
- **full-width match** for all nodes $g_i$ of $s$ with $i \in \mathbb{N}$, if $pg_j$ is a node of $u$, then $g_j$ is a node of $s$.

Now suppose further that $n$ in $u$ has $d \geq 0$ daughters. Given an ISL-$k$ tree transducer $\tau$, we use $\leftarrow \tau(u, b)$ to denote the set of all trees $t[\square_1 \leftarrow \tau]$, $\ldots$, $t[\square_d \leftarrow \tau]$ such that there is a rewrite rule $\langle s, a, t \rangle$ in $\tau$ that targets node $b$ in $u$. If this set is empty, $\leftarrow \tau(u, b)$ is undefined. For any $\Sigma$-tree $t$, we may simply write $\leftarrow \tau(t)$ instead of $\leftarrow \tau(t, \varepsilon)$.

For any tree language $L$, the transduction computed by $\tau$ in **synchronous mode** is $\leftarrow \tau(L) := \{ \langle i, o \rangle \mid i \in L, o \in \leftarrow \tau(i) \}$. A transduction is **synchronous input strictly $k$-local** (sISL-$k$) iff it can be computed by some ISL-$k$ transducer in synchronous mode.
The Big Linguistic Questions

- What are the laws that govern each structural level?
- **Why** are those the laws?
- How **complex** are these laws? How hard are they to compute?
- How are they **learned**?
- Do we find **typological gaps**, i.e. patterns that should exist but don’t appear in any language?
- What can we infer about human cognition?

The Opportunistic Program for Lazy Researchers Like Myself

- Stand on the shoulders of giants.
- Computer scientists have figured out a lot about complexity, so let’s apply their ideas to language.
The Big Linguistic Questions

- What are the laws that govern each structural level?
- **Why** are those the laws?
- How **complex** are these laws? How hard are they to compute?
- How are they **learned**?
- Do we find **typological gaps**, i.e. patterns that should exist but don’t appear in any language?
- What can we infer about human cognition?

The Opportunistic Program for Lazy Researchers Like Myself

- Stand on the shoulders of giants.
- Computer scientists have figured out a lot about complexity, so let’s apply their ideas to language.
A Mathematical Distinctness Theorem

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

regular $<$ context-free $<$ mildly context-sensitive $<$ ···

Phonology

Morphology

Syntax

Matches linguistic practice (despite attempts at unification, e.g. Government Phonology, DM, OT syntax)

A unified Theory of Everything is not on the linguistic horizon.
A Mathematical Distinctness Theorem

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

\[
\text{regular} < \text{context-free} < \text{mildly context-sensitive} < \cdots
\]

Kaplan and Kay (1994)

- Phonology

Morphology

Syntax

- Matches linguistic practice (despite attempts at unification, e.g. Government Phonology, DM, OT syntax)

- A unified Theory of Everything is not on the linguistic horizon.
A Mathematical Distinctness Theorem

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

- Regular $<$ Context-free $<$ Mildly context-sensitive $<$ \ldots

- Matches linguistic practice (despite attempts at unification, e.g. Government Phonology, DM, OT syntax)

- A unified Theory of Everything is not on the linguistic horizon.
A Mathematical Distinctness Theorem

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

regular $< \text{context-free} < \text{mildly context-sensitive} < \cdots$

Matches linguistic practice (despite attempts at unification, e.g. Government Phonology, DM, OT syntax)

A unified Theory of Everything is not on the linguistic horizon.
Cognitive Parallelism Hypothesis

- The postulated split is misleading.
- If we probe deeper, we find that
  - different modules are remarkably similar,
  - their dependencies are weaker than regular
    \[ \Rightarrow \text{subregular} \]
- Cognitive parallelism is \textit{empirically fertile}.

Take-Home Messages

1. Phonology and syntax show surprising subregular parallels.
   (Morphology and morphosemantics, too...)
2. Like every good theory, subregularity yields new generalizations and data.
Outline

1. Subregular Phonology: SL & TSL over Strings
   - Strictly Local (SL)
   - Tier-Based Strictly Local (TSL)

2. Subregular Syntax: SL & TSL over Trees
   - Formalizing Syntax
   - Merge is SL
   - Move is TSL
   - Islands ≡ Blocking

3. The Hidden Power of Subcategorization
SL & TSL: (T)ier-Based Strictly Local

- There are a variety of subregular classes to choose from.
- SL and TSL are among the weaker ones.
- They work well empirically.

(Tier-Based) Strictly Local Dependencies

- All patterns described by markedness constraints that are
  - inviolable,
  - locally bounded,
  - formalized as $n$-grams.
- Non-local dependencies are **local over tiers**. (Goldsmith 1976)
- **Linguistic core idea:**
  Dependencies are local over the right structure.
There are a variety of subregular classes to choose from. SL and TSL are among the weaker ones. They work well empirically.

All patterns described by markedness constraints that are:
- inviolable,
- locally bounded,
- formalized as n-grams.

Non-local dependencies are local over tiers.
(Goldsmith 1976)

Linguistic core idea:
Dependencies are local over the right structure.
Example: Word-Final Devoicing is SL-2

- 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

```
  *$\text{r a d }$  
  *$z$  
  *$v$  
  *$d$  
```
Example: Word-Final Devoicing is SL-2

- 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 is SL-2

- 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 is SL-2

- 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 is SL-2

- 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 is SL-2

- 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 is SL-2

- 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

```plaintext
* $r a d$

* $z$

* $v$

* $d$
```
Example: Word-Final Devoicing is SL-2

- 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**

<table>
<thead>
<tr>
<th>*z$</th>
<th>*v$</th>
<th>$r$</th>
<th>$a$</th>
<th>$t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>*$r\ a\ d$</td>
<td>*d$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example: Word-Final Devoicing is SL-2

- 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

\[
\begin{array}{c}
* z$
\end{array}
\begin{array}{c}
* v$
\end{array}
\begin{array}{c}
* d$
\end{array}
\begin{array}{c}
* r a d$
\end{array}
\begin{array}{c}
\$ r a t$
\end{array}
\]
Example: Word-Final Devoicing is SL-2

- 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

<table>
<thead>
<tr>
<th>* $r a d $</th>
<th>*$z$</th>
<th>*$v$</th>
<th>*$d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r a t $</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example: Word-Final Devoicing is SL-2

- 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 $  * $z$  $r a t$  
* $v$  
* $d$  
```
Example: Word-Final Devoicing is SL-2

- 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$  $ r a t $  * d$
```
Example: Intervocalic Voicing is SL-3

- Captured by forbidding voiceless segments between vowels
- **Description:** don’t have $V[\text{voice}]V$
- **Suppose:**
  - $[\text{voice}] = \{s,f\}$
  - $V = \{a,i,o,u\}$
- **Compiled out:** don’t have asa, aʃa, asi, aʃi, ...

Example: Northern Italian

* $a$ s o | a $
Example: Intervocalic Voicing is SL-3

- Captured by forbidding voiceless segments between vowels
- **Description:** don’t have $V[-\text{voice}]V$
- **Suppose:**
  - $[-\text{voice}] = \{s, f\}$
  - $V = \{a, i, o, u\}$
- **Compiled out:** don’t have *asa, aʃa, asi, aʃi, . . .*

Example: Northern Italian

* $a s o l a$ *
Example: Intervocalic Voicing is SL-3

- Captured by forbidding voiceless segments between vowels
- **Description:** don’t have $V[-\text{voice}]V$
- **Suppose:**
  - $[-\text{voice}] = \{s,f\}$
  - $V = \{a,i,o,u\}$
- **Compiled out:** don’t have $asa$, $afa$, $asi$, $afi$, . . .

Example: Northern Italian

* $\text{aso} \text{ l} \text{a}$
Example: Intervocalic Voicing is SL-3

- Captured by forbidding voiceless segments between vowels
- **Description:** don’t have $V[-\text{voice}]V$
- **Suppose:**
  - $[-\text{voice}] = \{s,f\}$
  - $V = \{a,i,o,u\}$
- **Compiled out:** don’t have asa, aʃa, asi, aʃi, . . .

Example: Northern Italian

* $\text{asa} \mid \text{ala}$
Example: Intervocalic Voicing is SL-3

- Captured by forbidding voiceless segments between vowels
- **Description:** don’t have \( V[\neg \text{voice}]V \)
- **Suppose:**
  - \( [\neg \text{voice}] = \{s,f\} \)
  - \( V = \{a,i,o,u\} \)
- **Compiled out:** don’t have \( \text{asa, a} \text{a, asi, a} \text{i, . . .} \)

Example: Northern Italian

\[
\star \text{ aso } | \text{ al } \quad \text{ azo } | \text{ al }
\]
Example: Intervocalic Voicing is SL-3

- Captured by forbidding voiceless segments between vowels
- **Description:** don’t have $V[-\text{voice}]V$
- **Suppose:**
  - $[-\text{voice}] = \{s, f\}$
  - $V = \{a, i, o, u\}$
- **Compiled out:** don’t have $asa, a\text{f}a, asi, a\text{f}i, \ldots$

---

Example: Northern Italian

* $a s o l a$

$azola$
Example: Intervocalic Voicing is SL-3

- Captured by forbidding voiceless segments between vowels
- **Description:** don’t have $V[−\text{voice}]V$
- **Suppose:**
  - $[−\text{voice}] = \{s,f\}$
  - $V = \{a,i,o,u\}$
- **Compiled out:** don’t have asa, aʃa, asi, aʃi, ...

Example: Northern Italian

```
*$ a s o l a $     $ a z o l a $
```
Example: Intervocalic Voicing is SL-3

- Captured by forbidding voiceless segments between vowels
- **Description:** don’t have $V[-\text{voice}]V$
- **Suppose:**
  - $[-\text{voice}] = \{s,f\}$
  - $V = \{a,i,o,u\}$
- **Compiled out:** don’t have $asa$, $afa$, $asi$, $afi$, ...
Example: Intervocalic Voicing is SL-3

- Captured by forbidding voiceless segments between vowels
- **Description:** don’t have $V[−\text{voice}]V$
- **Suppose:**
  - $[−\text{voice}] = \{s,f\}$
  - $V = \{a,i,o,u\}$
- **Compiled out:** don’t have *asa, aʃa, asi, aʃi, ...*

Example: Northern Italian

$$*$ aso la $ \quad $ azola $*$$
Example: Intervocalic Voicing is SL-3

- Captured by forbidding voiceless segments between vowels
- **Description:** don’t have V[−voice]V
- **Suppose:**
  - [−voice] = {s,f}
  - V = {a,i,o,u}
- **Compiled out:** don’t have asa, aʃa, asi, aʃi, ...

Example: Northern Italian

* $a s o \mid a$

$ a z o \mid a$
Example: Intervocalic Voicing is SL-3

- Captured by forbidding voiceless segments between vowels
- **Description:** don’t have $V[\text{—voice}]V$
- **Suppose:**
  - $[\text{—voice}] = \{s, \text{ʃ}\}$
  - $V = \{a, i, o, u\}$
- **Compiled out:** don’t have $asa$, $afa$, $asi$, $afi$, . . .

Example: Northern Italian

```
* $a\ s\ o\ l\ a\ $  $a\ z\ o\ l\ a\ $  $a\ +\ s\ o\ c\ i\ a\ l\ e\ $  
```
Example: Intervocalic Voicing is SL-3

- Captured by forbidding voiceless segments between vowels
- **Description:** don’t have $V[-\text{voice}]V$
- **Suppose:**
  - $[-\text{voice}] = \{s,f\}$
  - $V = \{a,i,o,u\}$
- **Compiled out:** don’t have *asa, aʃa, asi, aʃi*, . . .

**Example: Northern Italian**

```
* $a s o l a$        $a z o l a$        $a + s o c i a l e$
```
Example: Intervocalic Voicing is SL-3

- Captured by forbidding voiceless segments between vowels
- **Description:** don’t have $V[\neg \text{voice}]V$
- **Suppose:**
  - $[\neg \text{voice}] = \{s, f\}$
  - $V = \{a, i, o, u\}$
- **Compiled out:** don’t have $\text{asa, a}jf\text{a, asi, a}jf$, . . .

Example: Northern Italian

*$a s o l a*  $a z o l a*  $a + s o c i a l e*$
Example: Intervocalic Voicing is SL-3

- Captured by forbidding voiceless segments between vowels
- **Description:** don’t have $V[−voice]V$
- **Suppose:**
  - $[−voice] = \{s, f\}$
  - $V = \{a, i, o, u\}$
- **Compiled out:** don’t have *asa*, *afa*, *asi*, *afi*, …

### Example: Northern Italian

* $asa$ | $aza$ | $asa + sociale$
Example: Intervocalic Voicing is SL-3

- Captured by forbidding voiceless segments between vowels
- **Description:** don’t have $V[{-\text{voice}}]V$
- **Suppose:**
  - $[{-\text{voice}}] = \{s, f\}$
  - $V = \{a, i, o, u\}$
- **Compiled out:** don’t have $asa, afà, asi, afì, \ldots$

**Example: Northern Italian**

```
* $a s o l a$  $a z o l a$  $a + s o c i a l e$
```
Example: Intervocalic Voicing is SL-3

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

Example: Northern Italian

* $a \ s \ o \ l \ a \ $  
  $a \ z \ o \ l \ a \ $  
  $a + s o c i a l e \ $
Example: Intervocalic Voicing is SL-3

- Captured by forbidding voiceless segments between vowels
- **Description:** don’t have $V[-\text{voice}]V$
- **Suppose:**
  - $[-\text{voice}] = \{s,f\}$
  - $V = \{a,i,o,u\}$
- **Compiled out:** don’t have $asa$, $afa$, $asi$, $afi$, . . .

Example: Northern Italian

* $aslal$  $azol$  $asa+social$
Example: Intervocalic Voicing is SL-3

- Captured by forbidding voiceless segments between vowels
- **Description:** don’t have $V[-\text{voice}]V$
- **Suppose:**
  - $[-\text{voice}] = \{s, f\}$
  - $V = \{a, i, o, u\}$
- **Compiled out:** don’t have asa, af[a, asi, afi, ... 

---

**Example: Northern Italian**

\*
\$a\$ s o l a \$a\$ z o l a \$a + social e\$

---
Example: Intervocalic Voicing is SL-3

- Captured by forbidding voiceless segments between vowels
- **Description:** don’t have $V[\text{voice}]V$
- **Suppose:**
  - $[\text{voice}] = \{s, z\}$
  - $V = \{a, i, o, u\}$

- **Compiled out:** don’t have asa, aża, asi, afi, ...

**Example: Northern Italian**

* $\$a\$ s o l a \$  \$a\$ z o l a \$  \$a + s o c i a l e \$
A Problem: Samala Sibilant Harmony

- If multiple sibilants occur in the same word, they must all be \([+\text{anterior}] (s,z)\) or \([-\text{anterior}] (\mathfrak{f},\mathfrak{3})\).
- In other words: Don’t mix purple and teal.

\[
*\mathfrak{s}\mathfrak{f} \quad *s\mathfrak{3} \quad *z\mathfrak{f} \quad *z\mathfrak{3} \\
*\mathfrak{f}s \quad *s\mathfrak{3} \quad *\mathfrak{f}z \quad *z\mathfrak{3}
\]

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

Example: Samala (Applegate 1972)

\[
\begin{align*}
*\$ \text{has} & \times \text{i} \times \text{t} \times \text{i} \times \text{l} \times \text{a} \times \mathfrak{f} \times \$ \\
\$ \text{ha} & \mathfrak{f} \times \text{i} \times \text{n} \times \text{t} \times \text{i} \times \text{l} \times \text{a} \times \mathfrak{f} \times \$
\end{align*}
\]
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.

\[ *sʃ \quad *sʒ \quad *ʒʃ \quad *ʒʒ \quad *ʃs \quad *ʒʒ \quad *ʃʒ \quad *ʒz \]

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

Example: Samala (Applegate 1972)

\[ *$ h a s x i n t i l a w a f $ \]

\[ $ h a f x i n t i l a w a f $ \]
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.

\[
\begin{align*}
* s^ʃ & \quad * s^ʒ & \quad * z^ʃ & \quad * z^ʒ \\
* ŋ^ʃ & \quad * ŋ^ʒ & \quad * j^ʃ & \quad * j^ʒ
\end{align*}
\]

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

Example: Samala (Applegate 1972)

\[
\begin{align*}
* S \ x \ i \ n \ t \ i \ l \ a \ w \ a \ ŋ & S \\
S \ h \ a \ ŋ \ x \ i \ n \ t \ i \ l \ a \ w \ a \ ŋ & S
\end{align*}
\]
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.

\[
\begin{align*}
* s \hat{\imath} & \quad * s \hat{\imath} \\
* z \hat{\imath} & \quad * z \hat{\imath} \\
* \hat{\imath} s & \quad * \hat{\imath} s \\
* \hat{\imath} z & \quad * \hat{\imath} z
\end{align*}
\]

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

Example: Samala (Applegate 1972)

\[
\begin{align*}
* \$ h a s \times i n t i l a w a \hat{\imath} $ \\
\$ h a \hat{\imath} s \times i n t i l a w a \hat{\imath} $ \\
\$ h a \hat{\imath} s \times i n t i l a w a \hat{\imath} $
\end{align*}
\]
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.
  \[
  \ast sʃ \ast sʒ \ast zʃ \ast zʒ \\
  \ast ʃs \ast ʒs \ast ʃz \ast ʒz
  \]
- But: Sibilants can be arbitrarily far away from each other!

Example: Samala (Applegate 1972)

\[
\ast $ h a s x i n t i l a w a f $ \\
$ h a f x i n t i l a w a f $ \\
\ast $ s t a j a n o w o n w a f $ \\
\]
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.

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

Example: Samala (Applegate 1972)

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

$\text{haʃ} x i n t i l a w a ʃ$

*$_{s}$*

*$s_3*_{z}$*$z_3*_{ʃ}$*$ʒ$

*$ʃ_3*_{ʒ}*$ʒ*$ʃ_3*_{ʒ}$
Phonology Syntax Categories Conclusion

Making Long-Distance Dependencies Local

Let’s take a clue from phonology: create locality with **tiers**.
(Heinz et al. 2011)

Example: Samala Revisited

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

*$ has x int il aw aʃ*$

*$ haʃ x int il aw aʃ*$

Jeff Heinz
Let’s take a clue from phonology: create locality with **tiers**. (Heinz et al. 2011)

Example: Samala Revisited

1. Project sibilant tier

2. $^*s^\backslash, ^*s_3^\backslash, ^*z^\backslash, ^*z_3^\backslash, ^*s, ^*_3s, ^*_z, ^*_3z$

*$h a \, s x i n t i l a w a \backslash*$

*$h a \backslash x i n t i l a w a \backslash*$
Making Long-Distance Dependencies Local

Let’s take a clue from phonology: create locality with **tiers**.
(Heinz et al. 2011)

Example: Samala Revisited

1. Project sibilant tier
2. $^*s \_s, ^*s_3, ^*z \_z, ^*z_3, ^*_s, ^*_s, ^*z, ^*_z$

$$\begin{align*}
\text{\$} & \text{s} & \_ & \text{s} & \text{\$} \\
\text{\$} & \text{has} & \text{\$} & \text{\$} \\
\text{\$} & \text{has} & \text{xintilawa} & \text{\$} \\
\text{\$} & \text{ha} & \text{xintilawa} & \text{\$}
\end{align*}$$
Making Long-Distance Dependencies Local

Let’s take a clue from phonology: create locality with tiers.
(Heinz et al. 2011)

Example: Samala Revisited

1. Project sibilant tier


\[
\begin{array}{c}
\$ & s & \$ & & \$ & \$ & \$ \\
| & | & | & | & | & |
\ast \$ h a s \times i n t i l a w a \$ \\
\$ h a \$ & \times i n t i l a w a \$ & \$ \\
\end{array}
\]
Making Long-Distance Dependencies Local

Let’s take a clue from phonology: create locality with **tiers**.
(Heinz et al. 2011)

Example: Samala Revisited

1. Project sibilant tier

2. $^*$s$ \int$, $^*$s3, $^*$z$ \int$, $^*$z3, $^*$s, $^*$s3, $^*$z, $^*$z3

```
$ s 

* $ h a s x i n t i l a w a \int $   $ h a \int x i n t i l a w a \int $
```
Making Long-Distance Dependencies Local

Let’s take a clue from phonology: create locality with *tiers*.
(Heinz et al. 2011)

Example: Samala Revisited

1. Project sibilant tier
2. $^*_{s}�, ^*_{s}\underline{z}, ^*_{z}�, ^*_{z}\underline{z}, ^*_{s}s, ^*_{z}s, ^*_{s}z, ^*_{z}z$

\[ \begin{align*}
\text{\$ s } \text{\$} & \quad \text{\$ s } \text{\$} \\
\text{\$ h a s x i n t i l a w a } \text{\$} & \quad \text{\$ h a } \text{\$} \\
\end{align*} \]
Making Long-Distance Dependencies Local

Let’s take a clue from phonology: create locality with tiers.
(Heinz et al. 2011)

Example: Samala Revisited

1. Project sibilant tier

2. \*sʃ, \*sʒ, \*zʃ, \*zʒ, \*ʃs, \*ʒs, \*ʃz, \*ʒz

\[
\begin{align*}
\text{$ \quad s \quad \mid \mid \quad \int \quad \mid \mid \quad \text{ʃ}$} \\
\text{$ \quad *h \; a \; s \; x \; i \; n \; t \; i \; l \; a \; w \; a \; \int \text{ʃ}$} \\
\text{$ \quad \text{ʃ}$}
\end{align*}
\]
Making Long-Distance Dependencies Local

Let’s take a clue from phonology: create locality with tiers.
(Heinz et al. 2011)

Example: Samala Revisited

1. Project sibilant tier

2. $^{*}s\backslash, ^{*}s3, ^{*}z\backslash, ^{*}z3, ^{*}s, ^{*}3s, ^{*}z, ^{*}3z$

$\$ s  \$  \$  \$

$\$  \$$

$^{*}s h a s x i n t i l a w a \$  \$ h a \$$

$\$  \$$
Making Long-Distance Dependencies Local

Let’s take a clue from phonology: create locality with **tiers**.
(Heinz et al. 2011)

### Example: Samala Revisited

1. **Project sibilant tier**

2. \( *s, *s_3, *z, *z_3, *s, *s_3, *z, *z_3 \)

<table>
<thead>
<tr>
<th>Project sibilant tier</th>
<th>( *s, *s_3, *z, *z_3, *s, *s_3, *z, *z_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s$</td>
<td>$f$</td>
</tr>
<tr>
<td>$h a s$</td>
<td>$f$</td>
</tr>
<tr>
<td>$h a f x i n t i l a w a f$</td>
<td>$h a f x i n t i l a w a f$</td>
</tr>
</tbody>
</table>
Example: Blocking

- TSL can also handle blocking effects.
- **Slovenian sibilant harmony with blocking**
  1. \([-\text{ant}] \ldots [+\text{ant}]\) is illicit,
  2. unless \(t\) or \(d\) intervenes.
- **TSL-2 account**
  1. project all \([-\text{ant}], [+\text{ant}], t, \text{ and } d\)
  2. don’t have \([-\text{ant}] [+\text{ant}]\)

Example: Slovenian (Jurgec 2011; McMullin 2016)

* \$\text{s p i ñ}\$ \quad \$\text{z ñ d a ñ}\$
Example: Blocking

- TSL can also handle blocking effects.
- **Slovenian sibilant harmony with blocking**
  1. \([-\text{ant}] \ldots [+\text{ant}]\) is illicit,
  2. unless \(t\) or \(d\) intervenes.
- **TSL-2 account**
  1. project all \([-\text{ant}], [+\text{ant}], t,\) and \(d\)
  2. don’t have \([-\text{ant}] \ldots [+\text{ant}]\)

Example: Slovenian (Jurgec 2011; McMullin 2016)

\[
\begin{array}{c}
\text{S} & \text{p} & \text{i} & \text{S} \\
\text{z} & \text{i} & \text{d} & \text{a} & \text{S}
\end{array}
\]
Example: Blocking

- TSL can also handle blocking effects.
- **Slovenian sibilant harmony with blocking**
  1. $[–\text{ant}] \ldots [+\text{ant}]$ is illicit,
  2. unless t or d intervenes.
- **TSL-2 account**
  1. project all $[–\text{ant}]$, $[+\text{ant}]$, t, and d
  2. don’t have $[–\text{ant}]$ $[+\text{ant}]$

Example: Slovenian (Jurgec 2011; McMullin 2016)

$\begin{array}{c}
\verb|s| \mid \mid \mid \mid \mid \\
\verb|\star s p i \mid| \\
\verb|z \ d| \mid \mid \mid \\
\verb|z \ i \ d \ a| \mid \mid 
\end{array}$
SL and TSL for Phonology

- Linguistically natural
- Correct and efficient learning algorithm
  (Jardine and McMullin 2017)
- Low resource demands ⇒ cognitively plausible
- Captures wide range of phonotactic dependencies
- Cannot generate unattested patterns

Example: First-Last Harmony

- Harmony only holds between initial and final segments
- Linguistically plausible, yet unattested

$hasxintilawaf$ *$stajanowonwaf$
SL and TSL for Phonology

- Linguistically natural
- Correct and efficient learning algorithm
  (Jardine and McMullin 2017)
- Low resource demands ⇒ cognitively plausible
- Captures wide range of phonotactic dependencies
- Cannot generate unattested patterns

Example: First-Last Harmony

- Harmony only holds between initial and final segments
- Linguistically plausible, yet unattested

\[
\text{'-hash\times\text{intilawas}$ ~ *\text{'sta\text{jano\text{wono\text{wasa}}}$
\]
SL and TSL for Phonology

- Linguistically natural
- Correct and efficient learning algorithm
  (Jardine and McMullin 2017)
- Low resource demands \(\Rightarrow\) cognitively plausible
- Captures wide range of phonotactic dependencies
- Cannot generate unattested patterns

Example: First-Last Harmony

- Harmony only holds between initial and final segments
- Linguistically plausible, yet unattested

\[
\begin{align*}
\$ & \quad s \quad \lfloor \quad $ \\
| & \quad | \quad | \quad | \\
$ & \quad h \quad a \quad s \times i \quad n \quad t \quad i \quad l \quad a \quad w \quad a \quad \lfloor \quad $ \\
& \quad \ast \quad \$ \quad s \quad t \quad a \quad j \quad a \quad n \quad o \quad w \quad o \quad n \quad w \quad a \quad \lfloor \quad $
\end{align*}
\]
SL and TSL for Phonology

- Linguistically natural
- Correct and efficient learning algorithm
  (Jardine and McMullin 2017)
- Low resource demands $\Rightarrow$ cognitively plausible
- Captures wide range of phonotactic dependencies
- Cannot generate unattested patterns

Example: First-Last Harmony

- Harmony only holds between initial and final segments
- Linguistically plausible, yet unattested

```
$ s \int s$  $s \int s$
$has\timesintilawa\int$  $*s\inttajanowonwa\int$
```
Outline

1. Subregular Phonology: SL & TSL over Strings
   - Strictly Local (SL)
   - Tier-Based Strictly Local (TSL)

2. Subregular Syntax: SL & TSL over Trees
   - Formalizing Syntax
   - Merge is SL
   - Move is TSL
   - Islands ≡ Blocking

3. The Hidden Power of Subcategorization
Against the Received View

- This is about strings.
- Syntax is about trees!

- Phonology
  - regular < context-free < mildly context-sensitive < ...

- Morphology
  - Kaplan and Kay (1994)
  - Karttunen et al. (1992)

- Syntax
  - Shieber (1985)
Minimalist Grammars as a Computational Model of Syntax

- 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

<table>
<thead>
<tr>
<th>Chemistry</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>atoms</td>
<td>words</td>
</tr>
<tr>
<td>electrons</td>
<td>features</td>
</tr>
<tr>
<td>molecules</td>
<td>sentences</td>
</tr>
</tbody>
</table>
Choice of Representation: Derivation trees

CP

DP<sub>ω</sub>  C'

D  N  C  TP

I  I

which  man  do  T<sub>i</sub>  DP<sub>m</sub>  T'

I  I

-ed  Mary  t<sub>i</sub>  VP

V'

V  t<sub>w</sub>

kiss

Phrase Structure Tree
Choice of Representation: Derivation trees

Phrase Structure Tree

Derivation Tree

Mary [D− nom−] which [N+ D− wh−] man [N−]

kiss [D+ D+ V−]

do [T+ h+ wh+ C−]

- ed [V+ nom+ T− h−]

Which [N−]

Man [N−]

D− nom−
A Detailed Merge Example

(1) John [\textsubscript{VP} t lauged at Bill].

**Sequence of Merge steps:**

1. at selects DP (Bill)
2. laughed selects PP (at)
3. laughed selects DP (John)
A Detailed Merge Example

(1) John [VP t lauged at Bill].

Sequence of Merge steps:

1. at selects DP (Bill)
2. laughed selects PP (at)
3. laughed selects DP (John)
A Detailed Merge Example

(1) John [VP t lauged at Bill].

Sequence of Merge steps:

1. at selects DP (Bill)
2. laughed selects PP (at)
3. laughed selects DP (John)
A Detailed Merge Example

(1) John [\text{VP} \, t \, \text{lauged at Bill}].

**Sequence of Merge steps:**
1. *at* selects DP (*Bill*)
2. *laughed* selects PP (*at*)
3. *laughed* selects DP (*John*)
A Detailed Merge Example

(1) John [\text{VP} \ 't' \ laughed \ at \ Bill].

**Sequence of Merge steps:**

1. *at* selects DP (Bill)
2. *laughed* selects PP (at)
3. *laughed* selects DP (John)

**Merge Features**

Merge is controlled by

- selector features \(X^+\)
- category features \(X^-\)
Merge is SL-2

Merge is SL-2 over trees because we only need to look at

1. the mother and
2. its daughters
Merge is SL-2

Merge is SL-2 over trees because we only need to look at

1. the mother and
2. its daughters

laughed

\[
P^+ \ D^+ \ V^-
\]

\[
\text{John} \quad \text{at} \\
\text{D}^- \quad \text{D}^+ \quad \text{P}^-
\]

\[
\text{picture} \\
\text{N}^-
\]
Merge is SL-2

**Merge is SL-2 over trees** because we only need to look at

1. the mother and
2. its daughters

```
laughed
P⁺  D⁺  V⁻
```

```
John    at    picture
D⁻  D⁺  P⁻  N⁻
```
Merge is SL-2 over trees because we only need to look at
1 the mother and
2 its daughters

laughed
\[ P^+ \ D^+ \ V^- \]
John \[ D^- \] at \[ D^+ \ P^- \]
picture \[ N^- \]

laughed
\[ P^+ \ D^+ \ V^- \]
at
\[ D^+ \ P^- \]
Bill \[ D^- \]
Merge is SL-2

**Merge is SL-2 over trees** because we only need to look at

1. the mother and
2. its daughters

```
laughed
P⁺ D⁺ V⁻
```

```
John   at
D⁻    D⁺ P⁻
```

```
laughed
P⁺ D⁺ V⁻
```

```
at
D⁺ P⁻
```

```
picture
N⁻
```

```
Bill
D⁻
```
**Merge is SL-2**

**Merge is SL-2 over trees** because we only need to look at:

1. **the mother and**
2. **its daughters**

\[
\begin{array}{lll}
\text{laughed} & \text{John} & \text{at} \\
\text{P}^+ & \text{D}^+ & \text{V}^-
\end{array}
\]

\[
\begin{array}{lll}
\text{laughed} & \text{at} \\
\text{P}^+ & \text{D}^+ & \text{V}^-
\end{array}
\]

\[
\begin{array}{lll}
\text{laughed} & \text{Peter} & \text{at} & \text{John} \\
\text{P}^+ & \text{D}^+ & \text{V}^-
\end{array}
\]

\[
\begin{array}{lll}
\text{laughed} & \text{at} \\
\text{P}^+ & \text{D}^+ & \text{V}^-
\end{array}
\]

- **John**: \text{D}^- \quad \text{at} \quad \text{D}^+ \quad \text{P}^- \quad \text{picture} \quad \text{N}^- 
- **Bill**: \text{D}^- 
- **Bill**: \text{D}^- 
- **Bill**: \text{D}^- 
- **John**: \text{D}^-
Merge is SL-2

**Merge is SL-2 over trees** because we only need to look at

1. the mother and
2. its daughters

```
laughed
P⁺ D⁺ V⁻

John
D⁻

at
D⁺ P⁻

picture
N⁻

laughed
P⁺ D⁺ V⁻

at
D⁺ P⁻

Bill
D⁻

laughed
P⁺ D⁺ V⁻

at
D⁺ P⁻

Bill
D⁻

Peter
D⁻

John
D⁻
```
Movement is Local over Tree Tiers

laughed
P⁺ D⁺ V⁻

John at
D⁻    D⁺ P⁻

|      |
|      |
| Bill |
| D⁻   |
Movement is Local over Tree Tiers

\[
\text{laughed} \\
| \\
| \\
\text{John} \quad \text{at} \\
| \\
| \\
\text{Bill} \\
\text{D}^-
\]

\[
\text{D}^- \quad \text{nom}^- \quad \text{D}^+ \quad \text{P}^-
\]

\[
\text{V}^- \\
\text{D}^+ \quad \text{P}^+
\]

\[
\text{P}^+ \quad \text{D}^+ 
\]
Movement is Local over Tree Tiers

\[
\begin{array}{c}
\varepsilon \\
V^+ \text{ nom}^+ \ T^-
\end{array}
\]

\[
\begin{array}{c}
\text{laughed} \\
P^+ \ D^+ \ V^-
\end{array}
\]

\[
\begin{array}{c}
\text{John} \\
D^- \ \text{nom}^- \at \ D^+ \ P^-
\end{array}
\]

\[
\begin{array}{c}
\text{at} \\
\text{Bill} \\
D^-
\end{array}
\]
Movement is Local over Tree Tiers

nom-tier

\[ \varepsilon \]

\[ V^+ \text{ nom}^+ T^- \]

\[ \text{laughed} \]

\[ P^+ D^+ V^- \]

\[ \text{John} \]

\[ \text{at} \]

\[ D^- \text{ nom}^- \]

\[ D^+ P^- \]

\[ \text{Bill} \]

\[ D^- \]
Movement is Local over Tree Tiers

nom-tier

ε

V+ nom+ T−

ε

\[ \varepsilon \]

V+ nom+ T−

\[ \varepsilon \]

laughed

P+ D+ V−

\[ \varepsilon \]

John

D− nom−

\[ \varepsilon \]

at

D+ P−

\[ \varepsilon \]

Bill

D−
Movement is Local over Tree Tiers

nom-tier

\[ \varepsilon \]

\[ V^+ \text{ nom}^+ \text{ T}^- \]

\[ \varepsilon \]

\[ V^+ \text{ nom}^+ \text{ T}^- \]

John

\[ D^- \text{ nom}^- \]

\[ \varepsilon \]

\[ V^+ \text{ nom}^+ \text{ T}^- \]

\[ \text{laughed} \]

\[ P^+ \text{ D}^+ \text{ V}^- \]

John

\[ D^- \text{ nom}^- \]

\[ \text{at} \]

\[ D^- \text{ nom}^- \]

\[ D^+ \text{ P}^- \]

\[ \text{Bill} \]

\[ D^- \]
Movement is Local over Tree Tiers

nom-tier

ε
V+ nom+ T−

ε

| laughed
P+ D+ V−

John

D− nom−

D− nom− D+ P−

Bill

D−
## Tier with Multiple Movers

\[
\begin{array}{c}
\varepsilon \\
V^+ \text{ nom}^+ \ T^- \\
\text{complained} \\
C^+ \ D^+ \ V^- \\
\text{John} \quad \text{that} \\
D^- \text{ nom}^- \ T^+ \ C^- \\
\varepsilon \\
V^+ \text{ nom}^+ \ T^- \\
\text{slept} \\
D^+ \ V^- \\
\text{Bill} \\
D^- \text{ nom}^-
\end{array}
\]
Tier with Multiple Movers

nom-tier

\[
\begin{align*}
\epsilon \\
V^+ & \quad \text{nom}^+ & T^- \\
\text{complained} \\
C^+ & \quad D^+ & V^- \\
\hline
\text{John} & \quad \text{that} \\
D^- & \quad \text{nom}^- & T^+ & \quad C^- \\
\epsilon \\
V^+ & \quad \text{nom}^+ & T^- \\
\text{slept} \\
D^+ & \quad V^- \\
\text{Bill} \\
D^- & \quad \text{nom}^-
\end{align*}
\]
Tier with Multiple Movers

nom-tier

\[ \varepsilon \quad V^+ \text{nom}^+ \text{T}^- \]

John complained that slept Bill

\[ D^- \text{nom}^- \text{T}^+ \text{C}^- \]

\[ \varepsilon \quad V^+ \text{nom}^+ \text{T}^- \]

Bill

\[ D^- \text{nom}^- \]
Tier with Multiple Movers

nom-tier

\[ \varepsilon \]
\[ V^+ \text{ nom}^+ T^- \]

John
\[ D^- \text{ nom}^- \]

\[ \varepsilon \]
\[ V^+ \text{ nom}^+ T^- \]
complained
\[ C^+ D^+ V^- \]

that

\[ \varepsilon \]
\[ V^+ \text{ nom}^+ T^- \]
slept
\[ D^+ V^- \]

Bill
\[ D^- \text{ nom}^- \]
Tier with Multiple Movers

**nom-tier**

\[ \begin{align*}
\varepsilon & \quad V^+ \quad \text{nom}^+ \quad T^- \\
\varepsilon & \quad \text{complained} \\
\text{John} & \quad C^+ \quad D^+ \quad V^- \\
D^- & \quad \text{nom}^- \\
\end{align*} \]

\[ \begin{align*}
\varepsilon & \quad V^+ \quad \text{nom}^+ \quad T^- \\
\varepsilon & \quad \text{slept} \\
D^+ & \quad V^- \\
\text{Bill} & \quad D^- \quad \text{nom}^- \\
\end{align*} \]
Tier with Multiple Movers

nom-tier

ε
V⁺ nom⁺ T⁻

John
D⁻ nom⁻ V⁺ nom⁺ T⁻

ε

V⁺ nom⁺ T⁻

complained

C⁺ D⁺ V⁻

John
D⁻ nom⁻ T⁺ C⁻

ε

V⁺ nom⁺ T⁻

slept

D⁺ V⁻

Bill
D⁻ nom⁻
Tier with Multiple Movers

nom-tier

\[
\begin{align*}
&\varepsilon \\
&V^+ \text{ nom}^+ T^- \\
&\text{John} \\
&D^- \text{ nom}^- V^+ \text{ nom}^+ T^- \\
&\varepsilon
\end{align*}
\]

\[
\begin{align*}
&\varepsilon \\
&V^+ \text{ nom}^+ T^- \\
&\text{complained} \\
&C^+ D^+ V^- \\
&\text{John} \\
&D^- \text{ nom}^- T^+ C^- \\
&\varepsilon \\
&V^+ \text{ nom}^+ T^- \\
&slept \\
&D^+ V^- \\
&\text{Bill} \\
&D^- \text{ nom}^-
\end{align*}
\]
Tier with Multiple Movers

nom-tier

\[ \varepsilon \]
\[ V^+ \ nom^+ \ T^- \]
\[ \varepsilon \]
\[ D^- \ nom^- \ V^+ \ nom^+ \ T^- \]

John

Bill

\[ \varepsilon \]
\[ V^+ \ nom^+ \ T^- \]
\[ \varepsilon \]
\[ C^+ \ D^+ \ V^- \]

complained

that

\[ \varepsilon \]
\[ V^+ \ nom^+ \ T^- \]
\[ \varepsilon \]
\[ D^+ \ V^- \]

slept

Bill

D^- \ nom^-
Tier with Multiple Movers

nom-tier

John: $D^- nom^-$ $V^+ nom^+ T^-$

Bill: $D^- nom^-$ $V^+ nom^+ T^-$

$\varepsilon V^+ nom^+ T^-\varepsilon$

Complained $C^+ D^+ V^-\varepsilon$

John $\varepsilon$ that $D^- nom^-$ $T^+ C^-\varepsilon$

Slept $D^+ V^-\varepsilon$

Bill $D^- nom^-$
Separate Tier for Each Movement Type

\[
\begin{align*}
\text{did} & : T^+ \text{ wh}^+ \text{ C}^- \\
\text{complain} & : C^+ \text{ D}^+ \text{ V}^- \\
\text{slapped} & : D^+ \text{ D}^+ \text{ V}^- \\
\end{align*}
\]

\[
\begin{align*}
\text{John} & : D^- \text{ nom}^- \text{ T}^+ \text{ C}^- \\
\text{that} & : \epsilon \\
\text{Bill} & : D^- \text{ nom}^- \text{ D}^- \text{ wh}^- \\
\text{who} & : \epsilon \\
\end{align*}
\]
Separate Tier for Each Movement Type

nom-tier

\[ \varepsilon \]
\[ V^+ \text{ nom}^+ \text{ T}^- \]

\[ \text{John} \]
\[ D^- \text{ nom}^- \]
\[ V^+ \text{ nom}^+ \text{ T}^- \]
\[ \varepsilon \]
\[ \text{Bill} \]
\[ D^- \text{ nom}^- \]

\[ \text{did} \]
\[ T^+ \text{ wh}^+ \text{ C}^- \]
\[ \varepsilon \]
\[ V^+ \text{ nom}^+ \text{ T}^- \]
\[ \text{complain} \]
\[ C^+ \text{ D}^+ \text{ V}^- \]

\[ \text{John} \]
\[ D^- \text{ nom}^- \]
\[ T^+ \text{ C}^- \]
\[ \varepsilon \]
\[ V^+ \text{ nom}^+ \text{ T}^- \]
\[ \text{slapped} \]
\[ D^+ \text{ D}^+ \text{ V}^- \]

\[ \text{Bill} \]
\[ D^- \text{ nom}^- \]
\[ \text{who} \]
\[ D^- \text{ wh}^- \]
Separate Tier for Each Movement Type

did
T⁺ wh⁺ C⁻

ε
V⁺ nom⁺ T⁻

complain
C⁺ D⁺ V⁻

John
D⁻ nom⁻

ε
V⁺ nom⁺ T⁻

Bill
D⁻ nom⁻

did
T⁺ wh⁺ C⁻

ε
V⁺ nom⁺ T⁻

who
D⁻ wh⁻

John
D⁻ nom⁻

that
T⁺ C⁻

ε
V⁺ nom⁺ T⁻

slapped
D⁺ D⁺ V⁻

Bill
D⁻ nom⁻

who
D⁻ wh⁻
Move is TSL-2

- We now know how to construct movement tiers.
- Licit movement only creates tiers of a specific shape.
- **Move is TSL-2 over trees:**
  1. Every $f^-$ must have an $f^+$ mother.
  2. Every $f^+$ has exactly one $f^-$ among its daughters.

### Cognitive parallelism

<table>
<thead>
<tr>
<th></th>
<th>Phonology</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>SL</td>
<td>local dependencies</td>
<td>Merge</td>
</tr>
<tr>
<td>TSL</td>
<td>non-local dependencies</td>
<td>Move</td>
</tr>
</tbody>
</table>
Move is TSL-2

- We now know how to construct movement tiers.
- Licit movement only creates tiers of a specific shape.
- **Move is TSL-2 over trees:**
  1. Every $f^-$ must have an $f^+$ mother.
  2. Every $f^+$ has exactly one $f^-$ among its daughters.

### Cognitive parallelism

<table>
<thead>
<tr>
<th>Phonology</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>SL</td>
<td>local dependencies</td>
</tr>
<tr>
<td>TSL</td>
<td>non-local dependencies</td>
</tr>
</tbody>
</table>
Islands Come for Free

Two Fundamental Questions of Syntax

- Why do islands exist?
- Why do island exceptions exist?

A computational argument

1. Movement requires the power of TSL-2.
2. TSL-2 can model islands as blocking effects.
3. The cognitive ability for movement entails the cognitive ability for islands.
Islands Examples

(2) * Which car did John complain [because Bill damaged $t$].
(3) * Which car did John deny [the fact that Bill damaged $t$].
(4) Which car did John drive Mary crazy [while trying to fix $t$].
Islands Examples

(2) * Which car did John complain [because Bill damaged t].

(3) * Which car did John deny [the fact that Bill damaged t].

(4) Which car did John drive Mary crazy [while trying to fix t].

did
T^+  wh^+  C^−
|  
|  because
T^+  V^−
|  
|  which
D^−  wh^−
Islands Examples

(2) * Which car did John complain \([\text{because} \ \text{Bill damaged } t]\).
(3) * Which car did John deny \([\text{the} \ \text{fact} \ \text{that} \ \text{Bill damaged } t]\).
(4) Which car did John drive Mary crazy \([\text{while} \ \text{trying to fix } t]\).

\[
\begin{array}{c}
did \\
T^+ \ \text{wh}^+ \ C^- \\
| \\
\text{because} \\
T^+ \ V^\sim \\
| \\
\text{which} \\
D^- \ \text{wh}^- \\
\end{array}
\quad
\begin{array}{c}
did \\
T^+ \ \text{wh}^+ \ C^- \\
| \\
\text{fact} \\
C^+ \ N^- \\
| \\
\text{which} \\
D^- \ \text{wh}^- \\
\end{array}
\]
Islands Examples

(2) * Which car did John complain [because Bill damaged \( t \)].
(3) * Which car did John deny [the fact that Bill damaged \( t \)].
(4) Which car did John drive Mary crazy [while trying to fix \( t \)].

\[
\begin{align*}
\text{did} & \quad \text{T}^+ \quad \text{wh}^+ \quad \text{C}^- \\
\text{because} & \quad \text{T}^+ \quad \text{V}^{-} \\
\text{which} & \quad \text{D}^- \quad \text{wh}^- \\
\text{did} & \quad \text{T}^+ \quad \text{wh}^+ \quad \text{C}^- \\
\text{fact} & \quad \text{C}^+ \quad \text{N}^- \\
\text{which} & \quad \text{D}^- \quad \text{wh}^- \\
\text{did} & \quad \text{T}^+ \quad \text{wh}^+ \quad \text{C}^- \\
\text{which} & \quad \text{D}^- \quad \text{wh}^- 
\end{align*}
\]
Impossible Islands

- Islands arise when a blocker is projected onto a tier.
- Tier projection only considers lexical item itself, not its structural context.
- TSL-2 theory of islands hence rules out:
  - **Gang-up islands**
    “A mover can escape \( n \) islands, but not more than that.”
  - **Configurational islands**
    “An adjunct is an island iff it is inside an embedded clause.”
  - **Cowardly islands**
    “An adjunct is an island iff there are at least two adjuncts in the clause.”
  - **Rationed islands**
    “Only one adjunct per clause can be an island.”
  - **Discerning islands**
    “Adjuncts only block movers that contain an adjective.”
Outline

1. Subregular Phonology: SL & TSL over Strings
   - Strictly Local (SL)
   - Tier-Based Strictly Local (TSL)

2. Subregular Syntax: SL & TSL over Trees
   - Formalizing Syntax
   - Merge is SL
   - Move is TSL
   - Islands $\equiv$ Blocking

3. The Hidden Power of Subcategorization
### Hidden Power of Merge Features

#### A Confession
- Subregularity does not limit anything!
- Merge can do pretty much anything you want.

<table>
<thead>
<tr>
<th>Counting</th>
<th>every DP contains at least five LIs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symmetry closure</td>
<td>every reflexive c-commands its antecedent</td>
</tr>
<tr>
<td>Complement</td>
<td>sentence well-formed iff ill-formed in English</td>
</tr>
<tr>
<td>Boolean closure</td>
<td>sentence must obey either V2 or Principle A, unless there are less than 7 pronounced LIs</td>
</tr>
<tr>
<td>Domain blindness</td>
<td>a sentence is well-formed iff there are at least two words that display word-final devoicing</td>
</tr>
<tr>
<td>Islands</td>
<td>all the ones mentioned before smuggle movers out of islands</td>
</tr>
</tbody>
</table>
Why?

- Complex constraints can be compiled into the features.
- Once compiled in, they are enforced via Merge.
- It’s a generalized version of slash feature percolation.

Example: A grammar for modulo counting

\[
\begin{align*}
\varepsilon &:: O^+C^- \\
\| &
\end{align*}
\]

\[
\begin{align*}
\text{foo} &:: O^- \\
\| &
\end{align*}
\]

\[
\begin{align*}
\text{foo} &:: E^+O^- \\
\| &
\end{align*}
\]

\[
\begin{align*}
\text{foo} &:: O^+E^- \\
\| &
\end{align*}
\]

\[
\begin{align*}
\text{bar} &:: O^- \\
\| &
\end{align*}
\]

\[
\begin{align*}
\text{bar} &:: E^+O^- \\
\| &
\end{align*}
\]

\[
\begin{align*}
\text{bar} &:: O^+E^- \\
\| &
\end{align*}
\]

\[
\begin{align*}
\text{foo} &:: E^+O^- \\
\| &
\end{align*}
\]

\[
\begin{align*}
\text{bar} &:: O^+E^- \\
\| &
\end{align*}
\]

\[
\begin{align*}
\text{bar} &:: O^- \\
\| &
\end{align*}
\]
Local Feature Recoverability

- We need to restrict the power of Merge features, but how?
- Linguistic restrictions on categories don’t work (morphology, distribution, semantics)

**Feature Recoverability**

Merge features must be recoverable in an **SL** fashion.
Local Feature Recoverability

- We need to restrict the power of Merge features, but how?
- Linguistic restrictions on categories don’t work (morphology, distribution, semantics)

Feature Recoverability

Merge features must be recoverable in an SL fashion.

```
water :: D^+D^+V^-
  the :: N^+D^-
   gardeners :: N^- flowers :: D^-
  their :: N^+D^-
```
Local Feature Recoverability

- We need to restrict the power of Merge features, but how?
- Linguistic restrictions on categories don’t work (morphology, distribution, semantics)

Feature Recoverability

Merge features must be recoverable in an SL fashion.

water :: D⁺D⁺V⁻
the :: N⁺D⁻
their :: N⁺D⁻
gardeners :: N⁻
flowers :: D⁻

water
the
their
removal
gardeners
flowers
Local Feature Recoverability

- We need to restrict the power of Merge features, but how?
- Linguistic restrictions on categories don’t work (morphology, distribution, semantics)

Feature Recoverability

Merge features must be recoverable in an SL fashion.

Water :: $D^+D^+V^-$

The :: $N^+D^-$

Their :: $N^+D^-$

Gardeners :: $N^-$

Flowers :: $D^-$

Feature assignment

Removal
Modulo Counting is Not SL Recoverable

Can you determine the features of \texttt{foo}?

1. O$^+$ E$^-$
2. E$^+$ O$^-$

▶ No, that’s impossible.
▶ You need more than local information.
▶ Modulo counting is not SL recoverable.
Empirical Conjecture

SL-2 Recoverability Conjecture

For every lexical item $l$, the Merge features of $l$ are recoverable from feature-less dependency trees using only a window of size 2.
Implications and Open Issues

Implications

▶ We avoid tons of overgeneration.
▶ Heads only select for arguments, not arguments of arguments.
▶ Cognitive parallelism:
  Phonological feature inference equally complex (SL-1 or SL-2).

Open issues

▶ Needs to be tested across many languages
▶ Depends on theoretical assumptions
  ▶ distribution of empty heads
  ▶ lexical items fully inflected or bare roots?
▶ Move features cannot be inferred this way.
### Towards a Learning Algorithm for Minimalism

- Categories are a major hurdle for syntactic learning algorithms.
- Feature recoverability opens up a new strategy.

### A Learning Paradigm for Minimalist Syntax

<table>
<thead>
<tr>
<th></th>
<th><strong>Input</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>string (observed)</td>
</tr>
<tr>
<td></td>
<td>head-argument relations (basic semantic interpretation)</td>
</tr>
<tr>
<td></td>
<td>notion of feature recoverability (UG)</td>
</tr>
<tr>
<td>2</td>
<td>Construct feature-free dependency tree</td>
</tr>
<tr>
<td>3</td>
<td>Distributional learning of categories via recoverability (state merging)</td>
</tr>
<tr>
<td>4</td>
<td>Infer movement from string</td>
</tr>
</tbody>
</table>
Wrapping up: Concrete Results

- **Cognitive Parallelism**
  - Phonology and syntax are surprisingly similar.
  - SL and TSL play a central role in both.
  - Islands $\equiv$ Blocking
  - Both come for free

- **Specific Phenomenon: Subcategorization**
  - Linguists haven’t paid enough attention to subcategorization.
  - Subregular complexity makes strong predictions about categories.
Subregular Linguistics as Linguistics

- Subregular **linguistics**
  - What are the laws of grammar?
  - How complex are they?
  - Why are those the laws?
  - Are some analyses simpler than others?

- Interplay of theory and data:
  - new typological claims
  - deeper understanding of formalism through data
  - new empirical questions
  - unification of diverse data points
  - learnability
  - direct ties to cognition

- It’s just another tool. The more tools, the better!
Join the Program

Everybody has something to contribute!

- Do you have data that contradicts current predictions?
- Wanna add probabilities and gradience?
- In-depth analysis of specific phenomena
- grammar fragments
- artificial language learning experiments
- processing experiments
Follow Along (https://outde.xyz)

Shellshock

This semester I am teaching a seminar on computational syntax. It’s mostly on subregular syntax, but I started out with a discussion of CCG. CCG is noteworthy because it is a theory-rich approach that has managed to make major inroads into NLP. It would be cool if we could replicate this with MGs, but in order to do that you need a killer app. Subregular complexity might just be that because CCG doesn’t have a regular backbone, so it can’t have a subregular one either (more on that in a future post). CCG’s killer app was flexible constituency and a one-to-one mapping from syntax to semantics. You combine that with a corpus (CCGbank) and an efficient parsing algorithm (e.g. supertagging with A* parsing), and you have something that is both linguistically sophisticated and sufficiently fast and robust for practical applications. Anyways, this post collects some of my thoughts on flexible constituency and how it could be emulated in MGs. Spoiler: shells, lots and lots of shells.

We need a framework?

As you might know, Stony Brook hosted AMP, the American Meeting on Phonology, a week ago quite a while ago (yikes, almost November again). Jane Chandlee started off the conference with an invited talk on the subregular view of phonological mappings from underlying representations to surface forms. It was well received, but during the question period Bruce Hayes (no, not that Bruce Hayes) made a point that I found puzzling: “You need a framework!” Unfortunately I didn’t have time to ask Bruce afterwards what exactly he meant by that. But every conceivable interpretation I’ve come up with I vehemently disagree with, and I think Bruce’s demand for a framework stems from incorrectly applying the linguistic modus operandi to computational work.
Appendix
Joint work with Alëna Aksënova and Sophie Moradi.

It seems that morphotactics is also TSL.

(Aksënova et al. 2016)
Example: Unbounded *the day after*-Prefixation in German

- German has a prefix *über*.
- This prefix can be freely combined with *morgen* ‘tomorrow’.

Example

<table>
<thead>
<tr>
<th>morgen</th>
<th>tomorrow</th>
</tr>
</thead>
<tbody>
<tr>
<td>über+morgen</td>
<td>the day after tomorrow</td>
</tr>
<tr>
<td>(über+)n morgen</td>
<td>(the day after)n tomorrow</td>
</tr>
</tbody>
</table>

TSL Description

Tier: *über*, stem boundary +

Constraint

*über* must be prefix

Bigrams

* + *über*
Example: Unbounded *the day after*-Prefixation in German

- German has a prefix *über*.
- This prefix can be freely combined with *morgen* ‘tomorrow’.

<table>
<thead>
<tr>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>morgen</em></td>
</tr>
<tr>
<td><em>über</em> + <em>morgen</em></td>
</tr>
<tr>
<td>(<em>über</em>)^n <em>morgen</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TSL Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tier:</strong> <em>über</em>, stem boundary +</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Bigrams</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>über</em> must be prefix</td>
<td>* + <em>über</em></td>
</tr>
</tbody>
</table>
Example: Unbounded *the day after*-Prefixation in German

- German has a prefix *über*.
- This prefix can be freely combined with *morgen* ‘tomorrow’.

<table>
<thead>
<tr>
<th>Example</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>morgen</em></td>
<td>tomorrow</td>
<td></td>
</tr>
<tr>
<td><em>über</em> + <em>morgen</em></td>
<td>the day after tomorrow</td>
<td></td>
</tr>
<tr>
<td>(<em>über</em>)(^n) <em>morgen</em></td>
<td>(the day after)(^n) tomorrow</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TSL Description</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tier:</strong> <em>über</em>, stem boundary +</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Constraint</strong></td>
<td><em>über</em> must be prefix</td>
<td>$ <em>über</em> <em>über</em> +</td>
</tr>
<tr>
<td><strong>Bigrams</strong></td>
<td>* + <em>über</em></td>
<td>+ <em>über</em> $</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$ <em>über</em> <em>über</em> + <em>morgen</em> + <em>über</em> $</td>
</tr>
</tbody>
</table>
Example: Bounded *the day after*-Circumfixation in Ilocano

- Ilocano has a circumfix `ka- -an`.
- This prefix can be combined once with `bigát` ‘tomorrow’.

**Example**

<table>
<thead>
<tr>
<th><code>bigát</code></th>
<th>tomorrow</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ka+bigát+an</code></td>
<td>the day after tomorrow</td>
</tr>
<tr>
<td><em>(ka)^n+bigát+(an)^n</em></td>
<td><em>(the day after)^n</em> tomorrow</td>
</tr>
</tbody>
</table>

**TSL Description**

**Tier:** `ka`, `an`, stem boundary +

**Constraint**

- `ka` must be prefix
- `an` must be suffix
- `ka` before `an`
- no iteration
- no lonely affix

**Bigrams**

- `* + ka`
- `*an +`
- `*an ka`
- `*ka ka, *an an`
- `*ka ++ $, *$++ an`
Example: Bounded *the day after*-Circumfixation in Ilocano

- Ilocano has a circumfix ka- -an.
- This prefix can be combined once with bigát ‘tomorrow’.

Example

<table>
<thead>
<tr>
<th>bigát</th>
<th>tomorrow</th>
</tr>
</thead>
<tbody>
<tr>
<td>ka+bigát+an</td>
<td>the day after tomorrow</td>
</tr>
<tr>
<td>*(ka)^n+bigát+(an)^n</td>
<td>(the day after)^n tomorrow</td>
</tr>
</tbody>
</table>

TSL Description

**Tier:** ka, an, stem boundary +

**Constraint**
- ka must be prefix
- an must be suffix
- ka before an
- no iteration
- no lonely affix

**Bigrams**
- * + ka
- *an +
- *an ka
- *ka ka, *an an
- *ka ++ $, *$++ an
Example: Bounded *the day after*-Circumfixation in Ilocano

- Ilocano has a circumfix *ka- -an*.
- This prefix can be combined once with *bigát* ‘tomorrow’.

**Example**

<table>
<thead>
<tr>
<th>bigát</th>
<th>tomorrow</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>ka + bigát + an</em></td>
<td>the day after tomorrow</td>
</tr>
<tr>
<td><em>(ka)^n + bigát + (an)^n</em></td>
<td>(the day after)^n tomorrow</td>
</tr>
</tbody>
</table>

**TSL Description**

**Tier:** *ka, an, stem boundary +*

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Bigrams</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>ka</em> must be prefix</td>
<td>$ + ka</td>
</tr>
<tr>
<td><em>an</em> must be suffix</td>
<td>$ + an</td>
</tr>
<tr>
<td><em>ka</em> before <em>an</em></td>
<td>$ + ka ka, $ + an an</td>
</tr>
<tr>
<td>no iteration</td>
<td>$ + ka ++ $, $ + $ ++ an</td>
</tr>
<tr>
<td>no lonely affix</td>
<td>$ + an ka ka + bigát + $</td>
</tr>
</tbody>
</table>
Typological Gap: No Unbounded Circumfixation

- There seems to be no language with an affix that is freely iterable like German \textit{"uber}, and a circumfix like \textit{ka- -an} in Ilocano.

- Why this gap? Because the result would not be TSL!

**Explanation**

- The pattern would be $\textit{ka}^n + \textit{bigá}t + \textit{an}^n$.
- TSL cannot memorize exact numbers.
- All affixes would have to be visible in the same search window.
- But the window’s size is bounded, while the pattern is not.
Typological Gap: No Unbounded Circumfixation

- There seems to be no language with an affix that is
  - freely iterable like German "über", and
  - a circumfix like "ka- -an" in Ilocano.
- Why this gap? Because the result would not be TSL!

**Explanation**

- The pattern would be $ka^n + bigát + an^n$.
- TSL cannot memorize exact numbers.
- All affixes would have to be visible in the same search window.
- But the window’s size is bounded, while the pattern is not.
The importance of TSL for word structure seems to extend even into semantics.

**Case Study: Generalized Quantifiers**

A generalized quantifier may have a monomorphemic realization only if its quantifier language is TSL.
Quantifier Languages (van Bentheim 1986)

(5)  
a. Every student cheated.  
b. No student cheated.  
c. Some student cheated.  
d. Three students cheated.  

<table>
<thead>
<tr>
<th>students</th>
<th>John</th>
<th>Mary</th>
<th>Sue</th>
</tr>
</thead>
<tbody>
<tr>
<td>cheated</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>

| string  | Y    | N    | Y   |

- (5a): **False**, because the string contains a N
- (5b): **False**, because the string contains a Y
- (5c): **True**, because the string contains a Y
- (5d): **False**, because the string does not contain three Ys
Quantifier Languages (van Benthem 1986)

(5)  a. Every student cheated.
     b. No student cheated.
     c. Some student cheated.
     d. Three students cheated.

<table>
<thead>
<tr>
<th>students</th>
<th>John</th>
<th>Mary</th>
<th>Sue</th>
</tr>
</thead>
<tbody>
<tr>
<td>cheated</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>string</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
</tr>
<tr>
<td>N</td>
</tr>
<tr>
<td>Y</td>
</tr>
</tbody>
</table>

- (5a): False, because the string contains a N
- (5b): False, because the string contains a Y
- (5c): True, because the string contains a Y
- (5d): False, because the string does not contain three Ys
# TSL Descriptions for Quantifier Languages

<table>
<thead>
<tr>
<th>Quantifier</th>
<th>Constraint</th>
<th>$n$-grams</th>
<th>Tier</th>
</tr>
</thead>
<tbody>
<tr>
<td>every</td>
<td>$</td>
<td>N</td>
<td>= 0$</td>
</tr>
<tr>
<td>no</td>
<td>$</td>
<td>Y</td>
<td>= 0$</td>
</tr>
<tr>
<td>some</td>
<td>$</td>
<td>Y</td>
<td>\geq 1$</td>
</tr>
<tr>
<td>at least $n$</td>
<td>$</td>
<td>Y</td>
<td>\geq n$</td>
</tr>
<tr>
<td>at most $n$</td>
<td>$</td>
<td>Y</td>
<td>\leq n$</td>
</tr>
</tbody>
</table>

## Example

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$Y$</td>
<td>$Y$</td>
<td>___</td>
<td>____</td>
<td>____</td>
<td>____</td>
</tr>
<tr>
<td>___</td>
<td>___</td>
<td>____</td>
<td>____</td>
<td>____</td>
<td>____</td>
</tr>
<tr>
<td>$Y$</td>
<td>$N$</td>
<td>$Y$</td>
<td>$Y$</td>
<td>$Y$</td>
<td>$____</td>
</tr>
</tbody>
</table>

Some: *$$

At least 2: *$$, *$Y$$

At least 3: *$$, *$Y$$, *$YY$$

At most 2: *$YYY$

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$Y$</td>
<td>$Y$</td>
<td>___</td>
<td>____</td>
<td>____</td>
<td>____</td>
</tr>
<tr>
<td>___</td>
<td>___</td>
<td>____</td>
<td>____</td>
<td>____</td>
<td>____</td>
</tr>
<tr>
<td>$Y$</td>
<td>$N$</td>
<td>$Y$</td>
<td>$Y$</td>
<td>$Y$</td>
<td>$____</td>
</tr>
</tbody>
</table>

At most 2: *$YYY$

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$Y$</td>
<td>$Y$</td>
<td>$Y$</td>
<td>___</td>
<td>____</td>
<td>____</td>
</tr>
<tr>
<td>___</td>
<td>___</td>
<td>____</td>
<td>____</td>
<td>____</td>
<td>____</td>
</tr>
<tr>
<td>$Y$</td>
<td>$N$</td>
<td>$Y$</td>
<td>$Y$</td>
<td>$Y$</td>
<td>$____</td>
</tr>
</tbody>
</table>

At most 2: *$YYY$

**True**

**True**

**False**

**True**
## TSL Descriptions for Quantifier Languages

<table>
<thead>
<tr>
<th>Quantifier</th>
<th>Constraint</th>
<th>n-grams</th>
<th>Tier</th>
</tr>
</thead>
<tbody>
<tr>
<td>every</td>
<td>$</td>
<td>N</td>
<td>= 0$</td>
</tr>
<tr>
<td>no</td>
<td>$</td>
<td>Y</td>
<td>= 0$</td>
</tr>
<tr>
<td>some</td>
<td>$</td>
<td>Y</td>
<td>\geq 1$</td>
</tr>
<tr>
<td>at least n</td>
<td>$</td>
<td>Y</td>
<td>\geq n$</td>
</tr>
<tr>
<td>at most n</td>
<td>$</td>
<td>Y</td>
<td>\leq n$</td>
</tr>
</tbody>
</table>

### Example

<table>
<thead>
<tr>
<th>$Y$</th>
<th>$Y$</th>
<th>$Y$</th>
<th>$Y$</th>
<th>$Y$</th>
<th>$Y$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$Y$</td>
<td>$Y$</td>
<td>$N$</td>
<td>$Y$</td>
<td>$Y$</td>
<td></td>
</tr>
</tbody>
</table>

- **some** *$\$$*
- **at least 2** *$\$$, *$\$$* $Y$
- **at least 3** *$\$$, *$\$$* $Y$, *$\$$* $YY$ *
- **at most 2** *$YYY$*

- True
- True
- False
- True
## TSL Descriptions for Quantifier Languages

<table>
<thead>
<tr>
<th>Quantifier</th>
<th>Constraint</th>
<th>$n$-grams</th>
<th>Tier</th>
</tr>
</thead>
<tbody>
<tr>
<td>every</td>
<td>$</td>
<td>N</td>
<td>= 0$</td>
</tr>
<tr>
<td>no</td>
<td>$</td>
<td>Y</td>
<td>= 0$</td>
</tr>
<tr>
<td>some</td>
<td>$</td>
<td>Y</td>
<td>\geq 1$</td>
</tr>
<tr>
<td>at least $n$</td>
<td>$</td>
<td>Y</td>
<td>\geq n$</td>
</tr>
<tr>
<td>at most $n$</td>
<td>$</td>
<td>Y</td>
<td>\leq n$</td>
</tr>
</tbody>
</table>

### Example

<table>
<thead>
<tr>
<th>Characters</th>
<th>Constraints</th>
<th>Truth</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Y Y Y$</td>
<td>some</td>
<td><strong>True</strong></td>
</tr>
<tr>
<td>$Y Y Y Y$</td>
<td>at least 2</td>
<td><strong>True</strong></td>
</tr>
<tr>
<td>$Y Y Y Y$</td>
<td>at least 3</td>
<td><strong>False</strong></td>
</tr>
<tr>
<td>$Y Y N Y$</td>
<td>at most 2</td>
<td><strong>True</strong></td>
</tr>
</tbody>
</table>
Overview of Quantifier Languages

If a quantifier language is **not TSL**, then its quantifier **cannot be monomorphemic** in any language.

<table>
<thead>
<tr>
<th>Quantifier</th>
<th>TSL?</th>
<th>Tier</th>
<th>Mono.</th>
<th>(Paperno 2011)</th>
</tr>
</thead>
<tbody>
<tr>
<td>every</td>
<td>yes</td>
<td>none</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>no</td>
<td>yes</td>
<td>none</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>some</td>
<td>yes</td>
<td>Y</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>(at least) two</td>
<td>yes</td>
<td>Y</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>(at most) two</td>
<td>yes</td>
<td>Y</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>not all</td>
<td>yes</td>
<td>N</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>all but one</td>
<td>yes</td>
<td>N</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>even number</td>
<td>no</td>
<td></td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>prime number</td>
<td>no</td>
<td></td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>infinitely many</td>
<td>no</td>
<td></td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>most</td>
<td>no</td>
<td></td>
<td>???</td>
<td></td>
</tr>
</tbody>
</table>
Overview of Quantifier Languages

If a quantifier language is **not TSL**, then its quantifier **cannot be monomorphemic** in any language.

<table>
<thead>
<tr>
<th>Quantifier</th>
<th>TSL?</th>
<th>Tier</th>
<th>Mono.</th>
</tr>
</thead>
<tbody>
<tr>
<td>every</td>
<td>yes</td>
<td>none</td>
<td>yes</td>
</tr>
<tr>
<td>no</td>
<td>yes</td>
<td>none</td>
<td>yes</td>
</tr>
<tr>
<td>some (at least) two</td>
<td>yes</td>
<td>Y</td>
<td>yes</td>
</tr>
<tr>
<td>not all</td>
<td>yes</td>
<td>N</td>
<td>no</td>
</tr>
<tr>
<td>all but one</td>
<td>yes</td>
<td>N</td>
<td>no</td>
</tr>
<tr>
<td>even number</td>
<td>no</td>
<td></td>
<td>no</td>
</tr>
<tr>
<td>prime number</td>
<td>no</td>
<td></td>
<td>no</td>
</tr>
<tr>
<td>infinitely many</td>
<td>no</td>
<td></td>
<td>no</td>
</tr>
<tr>
<td>most</td>
<td>no</td>
<td></td>
<td>???</td>
</tr>
</tbody>
</table>

(Paperno 2011)
Overview of Quantifier Languages

If a quantifier language is not TSL, then its quantifier cannot be monomorphemic in any language.

<table>
<thead>
<tr>
<th>Quantifier</th>
<th>TSL?</th>
<th>Tier</th>
<th>Mono.</th>
</tr>
</thead>
<tbody>
<tr>
<td>every</td>
<td>yes</td>
<td>none</td>
<td>yes</td>
</tr>
<tr>
<td>no</td>
<td>yes</td>
<td>none</td>
<td>yes</td>
</tr>
<tr>
<td>some</td>
<td>yes</td>
<td>Y</td>
<td>yes</td>
</tr>
<tr>
<td>(at least) two</td>
<td>yes</td>
<td>Y</td>
<td>yes</td>
</tr>
<tr>
<td>(at most) two</td>
<td>yes</td>
<td>Y</td>
<td>yes</td>
</tr>
<tr>
<td>not all</td>
<td>yes</td>
<td>N</td>
<td>no</td>
</tr>
<tr>
<td>all but one</td>
<td>yes</td>
<td>N</td>
<td>no</td>
</tr>
<tr>
<td>even number</td>
<td>no</td>
<td></td>
<td>no</td>
</tr>
<tr>
<td>prime number</td>
<td>no</td>
<td></td>
<td>no</td>
</tr>
<tr>
<td>infinitely many</td>
<td>no</td>
<td></td>
<td>no</td>
</tr>
<tr>
<td>most</td>
<td>no</td>
<td></td>
<td>???</td>
</tr>
</tbody>
</table>

(Paperno 2011)
Overview of Quantifier Languages

If a quantifier language is **not TSL**, then its quantifier **cannot be monomorphemic** in any language.

<table>
<thead>
<tr>
<th>Quantifier</th>
<th>TSL?</th>
<th>Tier</th>
<th>Mono.</th>
</tr>
</thead>
<tbody>
<tr>
<td>every</td>
<td>yes</td>
<td>none</td>
<td>yes</td>
</tr>
<tr>
<td>no</td>
<td>yes</td>
<td>none</td>
<td>yes</td>
</tr>
<tr>
<td>some</td>
<td>yes</td>
<td>Y</td>
<td>yes</td>
</tr>
<tr>
<td>(at least) two</td>
<td>yes</td>
<td>Y</td>
<td>yes</td>
</tr>
<tr>
<td>(at most) two</td>
<td>yes</td>
<td>Y</td>
<td>yes</td>
</tr>
<tr>
<td>not all</td>
<td>yes</td>
<td>N</td>
<td>no</td>
</tr>
<tr>
<td>all but one</td>
<td>yes</td>
<td>N</td>
<td>no</td>
</tr>
<tr>
<td>even number</td>
<td>no</td>
<td></td>
<td>no</td>
</tr>
<tr>
<td>prime number</td>
<td>no</td>
<td></td>
<td>no</td>
</tr>
<tr>
<td>infinitely many</td>
<td>no</td>
<td></td>
<td>no</td>
</tr>
<tr>
<td>most</td>
<td>no</td>
<td></td>
<td>???</td>
</tr>
</tbody>
</table>

(Paperno 2011)
**The Case of *most***

There is good semantic evidence that “most” is internally complex and hence **not monomorphemic.**  (Hackl 2009)

<table>
<thead>
<tr>
<th>Quantifier</th>
<th>TSL?</th>
<th>Tier</th>
<th>Mono.</th>
</tr>
</thead>
<tbody>
<tr>
<td>every</td>
<td>yes</td>
<td>none</td>
<td>yes</td>
</tr>
<tr>
<td>no</td>
<td>yes</td>
<td>none</td>
<td>yes</td>
</tr>
<tr>
<td>some</td>
<td>yes</td>
<td>Y</td>
<td>yes</td>
</tr>
<tr>
<td>(at least) two</td>
<td>yes</td>
<td>Y</td>
<td>yes</td>
</tr>
<tr>
<td>(at most) two</td>
<td>yes</td>
<td>Y</td>
<td>yes</td>
</tr>
<tr>
<td>not all</td>
<td>yes</td>
<td>N</td>
<td>no</td>
</tr>
<tr>
<td>all but one</td>
<td>yes</td>
<td>N</td>
<td>no</td>
</tr>
<tr>
<td>even number</td>
<td>no</td>
<td></td>
<td>no</td>
</tr>
<tr>
<td>prime number</td>
<td>no</td>
<td></td>
<td>no</td>
</tr>
<tr>
<td>infinitely many</td>
<td>no</td>
<td></td>
<td>no</td>
</tr>
<tr>
<td>most</td>
<td>no</td>
<td></td>
<td>no</td>
</tr>
</tbody>
</table>


References II


