Tiers and Relativized Locality Across Language Modules

Thomas Graf
Aniello De Santo
Jon Rawski
Alëna Aksënova
Hossep Dolutian
Sedigheh Moradi
Hyunah Baek
Suji Yang
Jeffrey Heinz

Stony Brook University
aniello.desanto@stonybrook.edu
https://aniellodesanto.github.io/

Parallels Between Phonology & Syntax
Amsterdam, July 9, 2018
The Subregular Group @ SBU

Jeff Heinz

Thomas Graf

Alëna Aksënova

Hyunah Baek

Hossep Dolutian

Sedigheh Moradi

Jon Rawski

Suji Yang
The Elevator Pitch

Parallels between phonology and syntax?

- What would a computational linguist tell you?
- What will I show you today?
  They are fundamentally similar!

The Take-Home Message

- Two kind of dependencies: local and non-local
- The core mechanisms are the same cross-domain, over the respective structural representations.
- Relativized locality plays a major role
The Elevator Pitch

Parallels between phonology and syntax?

▶ What would a computational linguist tell you?
   Probably none!

▶ What will I show you today?
   They are fundamentally similar!

The Take-Home Message

▶ Two kind of dependencies: local and non-local
▶ The core mechanisms are the same cross-domain, over the respective structural representations.
▶ Relativized locality plays a major role
The Elevator Pitch

Parallels between phonology and syntax?

▶ What would a computational linguist tell you?
  Probably none!

▶ What will I show you today?
  They are fundamentally similar!

The Take-Home Message

▶ Two kind of dependencies: local and non-local
▶ The core mechanisms are the same cross-domain, over the respective structural representations.
▶ Relativized locality plays a major role
Outline

1 Local Dependencies
   ▶ In Phonology
   ▶ In Syntax

2 Non-local Dependencies
   ▶ In Phonology
   ▶ In Syntax

A methodological note:
   ▶ Only phonotactics considered (no input-output mappings)
   ▶ Minimalist Grammars (Stabler 1997) as a model of syntax
   ▶ Formal language theory as a tool to assess parallelisms
Outline

1 Local Dependencies
   ▶ In Phonology
   ▶ In Syntax

2 Non-local Dependencies
   ▶ In Phonology
   ▶ In Syntax

A methodological note:
   ▶ Only phonotactics considered (no input-output mappings)
   ▶ Minimalist Grammars (Stabler 1997) as a model of syntax
   ▶ Formal language theory as a tool to assess parallelisms
Local Dependencies in Phonology

1 Word-final devoicing
Forbid voiced segments at the end of a word

(1) a. *rad
    b. rat

1 Intervocalic voicing
Forbid voiceless segments in between two vowels

(2) a. *faser
    b. fazer

These patterns can be described by strictly local (SL) constraints.
Local Dependencies in Phonology

1. **Word-final devoicing**
   Forbid voiced segments at the end of a word
   
   (1)  
   a. *rad
   b. rat

1. **Intervocalic voicing**
   Forbid voiceless segments in between two vowels
   
   (2)  
   a. *faser
   b. fazer

These patterns can be described by **strictly local** (SL) constraints.
Local Dependencies in Phonology are SL

Example: Word-final devoicing

- Forbid voiced segments at the end of a word: \( *[+\text{voice}] \$
- **German**: \(*z*, *v*, *d* \( \$ \) (\( \$ \) = word edge).

\[
\begin{align*}
\$ r a d \$ & \quad \$ r a t \$
\end{align*}
\]

Example: Intervocalic voicing

- Forbid voicess segments in-between two vowels: \( \text{V}[-\text{voice}]\text{V} \)
- **German**: \(*\text{ase}, *\text{ise}, *\text{ese}, *\text{isi}, \ldots*

\[
\begin{align*}
\$ f a s e r \$ & \quad \$ f a z e r \$
\end{align*}
\]
Local Dependencies in Phonology are SL

Example: Word-final devoicing

- Forbid voiced segments at the end of a word: *[+voice]*
- German: *z*, *v*, *d* ($ = word edge)$


Example: Intervocalic voicing

- Forbid voiced segments in-between two vowels: *[voice]*
- German: *ase, ise, ese, isi*, ...

$\text{fa} \text{sel}$ $\text{fa} \text{zer}$
Local Dependencies in Phonology are SL

Example: Word-final devoicing

- Forbid voiced segments at the end of a word: *+[+voice]$
- **German**: *z$, *v$, *d$ ($ = word edge).

* $r$ a d $ \text{ok} $ $r$ a t $

Example: Intervocalic voicing

- Forbid voicess segments in-between two vowels: *V[-voice]V
- **German**: *ase, *ise, *ese, *isi, ...

* f a s e r $ \text{ok} $ f a z e r $
What about Syntax?

We need a model for 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

Local dependencies in syntax

- Merge is a feature-driven operation:
  category feature $N^-, D^-$, ...
  selector feature $N^+, D^+$, ...
- Subcategorization as formalized by Merge is strictly local.
What about Syntax?

We need a model for 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

**Local dependencies in syntax**

- Merge is a **feature-driven** operation:
  category feature $N^-$, $D^-$, ...
  selector feature $N^+$, $D^+$, ...
- Subcategorization as formalized by Merge is **strictly local**.
Local Dependencies in Syntax

Merge is a **feature-driven** operation:

- category feature $N^-, D^-$, ...
- selector feature $N^+, D^+$, ...

's cat

$N^+, D^+, D^-, N^-$
Local Dependencies in Syntax

Merge is a **feature-driven** operation:

- category feature $N^-, D^-, ...$
- selector feature $N^+, D^+, ...$

![Diagram of Merge operation with features]

$$\text{Merge} \quad \text{'s} \quad \text{cat} \quad N^+ \quad D^+ \quad D^- \quad N^-$$
Local Dependencies in Syntax

Merge is a **feature-driven** operation:

- category feature $N^-, D^-, ...$
- selector feature $N^+, D^+, ...$

```
Mary
  D^-

Merge

's
N^+, D^+, D^-
```

```
cat
N^-
```
Local Dependencies in Syntax

Merge is a **feature-driven** operation:

- category feature $N^-, D^-, ...$
- selector feature $N^+, D^+, ...$

![Diagram of Merge process](image-url)
Merge is SL (Graf 2012a)

SL constraints on Merge

- We lift constraints from string $n$-grams to tree $n$-grams
- We get SL constraints over subtrees.

\[ *\text{Merge} \]
\[
\begin{array}{c}
  a \\
  b \\
\end{array}
\]
\[
\begin{array}{c}
  X^+ \\
  D^- \\
  \neg X^- \\
\end{array}
\]
Merge is SL (Graf 2012a)

- We lift constraints from string *n*-grams to tree *n*-grams.
- We get SL constraints over subtrees.
Interim Summary

<table>
<thead>
<tr>
<th>Local Data Structure</th>
<th>Local</th>
<th>Data Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Phonology</strong></td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td><strong>Syntax</strong></td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

Local phenomena modeled by $n$-grams of bounded size:

- computationally very simple
- learnable from positive examples of strings/trees
- plausible cognitive requirements
Interim Summary

<table>
<thead>
<tr>
<th>Local</th>
<th>Data Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phonology</td>
<td>SL</td>
</tr>
<tr>
<td>Syntax</td>
<td>SL</td>
</tr>
</tbody>
</table>

Local phenomena modeled by $n$-grams of bounded size:

- computationally very simple
- learnable from positive examples of strings/trees
- plausible cognitive requirements
Interim Summary

<table>
<thead>
<tr>
<th></th>
<th>Local</th>
<th>Non-local</th>
<th>Data Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phonology</td>
<td>SL</td>
<td>?</td>
<td>Strings</td>
</tr>
<tr>
<td>Syntax</td>
<td>SL</td>
<td>?</td>
<td>Trees</td>
</tr>
</tbody>
</table>

Local phenomena modeled by $n$-grams of bounded size:

- computationally very simple
- learnable from positive examples of strings/trees
- plausible cognitive requirements
Unbounded Dependencies in Phonology

- **Samala Sibilant Harmony**
  Sibilants must not disagree in anteriority.
  (Applegate 1972)

  \[(3) \begin{align*}
  a. & \quad \ast \text{hasxintilawaʃ} \\
  b. & \quad \ast \text{haʃxintilawas} \\
  c. & \quad \text{haʃxintilawaʃ}
  \end{align*}\]

- **Unbounded Tone Plateauing in Luganda (UTP)**
  No L may occur within an interval spanned by H.
  (Hyman 2011)

  \[(4) \begin{align*}
  a. & \quad \text{LHLLLL} \\
  b. & \quad \text{LLLLLH} \\
  c. & \quad \ast \text{LHLLHL} \\
  d. & \quad \text{LHHHHL}
  \end{align*}\]
Unbounded Dependencies Are Not SL

- **Samala Sibilant Harmony**
  Sibilants must not disagree in anteriority.
  (Applegate 1972)

  (5)  
  a.  * hasxintilawaʃ 
  b.  * haʃxintilawas 
  c.  haʃxintilawaʃ 

Example: Samala

* $hasxintilawaʃ$ 

$haʃxintilawaʃ$
Unbounded Dependencies Are Not SL

- **Samala Sibilant Harmony**
  Sibilants must not disagree in anteriority.
  (Applegate 1972)

(5)  
  a.  * has\textit{xintilawa}ʃ
  b.  * haʃ\textit{xintilawas}
  c.  haʃ\textit{xintilawa}ʃ

Example: Samala

\begin{center}
  * $h a s x i n t i l a w a ʃ$
  \hspace{2cm}
  $h a ʃ x i n t i l a w a ʃ$
\end{center}
Unbounded Dependencies Are Not SL

- **Samala Sibilant Harmony**
  Sibilants must not disagree in anteriority.  
  (Applegate 1972)

  (5)  
  a. * hasxintilawaʃ  
  b. * haʃxintilawas  
  c. haʃxintilawaʃ

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$
Unbounded Dependencies Are Not SL

- **Samala Sibilant Harmony**
  Sibilants must not disagree in anteriority.
  (Applegate 1972)

(5) a. * has xintilawaʃ
b. * haʃxintilawas
c. haʃxintilawaʃ

Example: Samala

*$ haʃ{s x i n t i l a w a f}$

*$ haʃs x i n t i l a w a f*$

*$ haʃʃ x i n t i l a w a f*$
Unbounded Dependencies Are Not SL

- **Samala Sibilant Harmony**
  Sibilants must not disagree in anteriority.
  
  (Applegate 1972)

  (5) a. * hasxintilawaʃ
  b. * haʃxintilawas
  c. haʃxintilawaʃ

Example: Samala

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

$\text{s h a x i n t i l a w a f} \text{a}$

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

  * $\text{s t a j a n o w o n w a f} \text{a}$*
Unbounded Dependencies Are Not SL

- **Samala Sibilant Harmony**
  Sibilants must not disagree in anteriority.
  (Applegate 1972)

  \[(5)\]
  \[
  \begin{align*}
  a. \ & \ast \text{has} \text{xintilawa} \\
  b. \ & \ast \text{ha} \text{xintilawas} \\
  c. \ & \text{ha} \text{xintilawa} \\
  \end{align*}
  \]

**Example: Samala**

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

  \[
  \ast \text{s t a j a n o w o n w a} \]

  "hsxintilawa"
Locality Over Tiers

- Sibilants can be arbitrarily far away from each other!
- **Problem**: SL limited to locality domains of size $n$;

**Tier-based Strictly Local (TSL) Grammars (Heinz et al. 2011)**

- Projection of selected segments on a tier $T$;
- Strictly local constraints over $T$ determine wellformedness;
- Unbounded dependencies are local over tiers.
Locality Over Tiers

- Sibilants can be arbitrarily far away from each other!
- **Problem**: SL limited to locality domains of size $n$;

Tier-based Strictly Local (TSL) Grammars (Heinz et al. 2011)

- Projection of selected segments on a tier $T$;
- Strictly local constraints over $T$ determine wellformedness;
- Unbounded dependencies are local over **tiers**.
Unbounded Dependencies are TSL

Let’s revisit Samala Sibilant Harmony

(6)  a. * hasxintilawaʃ
    b. * haʃxintilawas
    c. haʃxintilawaʃ

What do we need to project? [+strident]

What do we need to ban? * [+ant][−ant], * [−ant][+ant]

Example: TSL Samala

* $ha{s}x{\text{intilaw}}{ʃ}$
ok $haʃ{x}{\text{intilaw}}{ʃ}$
Unbounded Dependencies are TSL

- Let’s revisit Samala Sibilant Harmony

  (6)  
  a. * hasxintilawaʃ 
  b. * haʃxintilawas 
  c. haʃxintilawaʃ 

- What do we need to project? [±strident] 
- What do we need to ban? * [+ant][−ant], * [−ant][+ant]
Unbounded Dependencies are TSL

Let’s revisit Samala Sibilant Harmony

(6)  

a.  * hasxintilawaf

b.  * hafxintilawas

c.  hafxintilawaf

What do we need to project? [+strident]

What do we need to ban? * [+ant] [−ant], * [−ant] [+ant]

Example: TSL Samala

* $NASDAQ$  $ok$ $NASDAQ$
Unbounded Dependencies are TSL

Let’s revisit Samala Sibilant Harmony

(6)  
   a. * hasxintilawaʃ
   b. * haʃxintilawas
   c. haʃxintilawaʃ

What do we need to project? [+strident]
What do we need to ban? *+[+ant][−ant], *[−ant][+ant]

Example: TSL Samala

.................................

* $\text{has}_\text{xintilawa}_\text{ʃ}$

  ok $\text{ha}_\text{ʃxintilaw}_\text{ʃ}$
Unbounded Dependencies are TSL

- Let’s revisit Samala Sibilant Harmony
  
  6. a. \( * \text{has}x\text{intilawa} \)
  
  b. \( * \text{ha}x\text{intilawa} \)
  
  c. \( \text{ha}x\text{intilawa} \)

- What do we need to project? \([+\text{strident}]\)
- What do we need to ban? \(*[+\text{ant}][-\text{ant}], *[−\text{ant}][+\text{ant}]\)

Example: TSL Samala

\[
\text{S} \\
\text{.................................................} \\
* \text{hSxintilawa} \quad \text{ok} \text{hfxintilawa} \\
\]
Unbounded Dependencies are TSL

- Let’s revisit Samala Sibilant Harmony

(6)   a. * hasxintilawaʃ
      b. * haʃxintilawaš
      c. haʃxintilawaʃ

- What do we need to project? [+strident]
- What do we need to ban? *+[−ant][−ant], *−[−ant]+[ant]

Example: TSL Samala

```
  s

* $ has$ x int il aw $  ok $ haʃ x int il aw $  
```
Unbounded Dependencies are TSL

- Let’s revisit Samala Sibilant Harmony
  
  (6)  
  a.  * hasxintilawas
  
  b.  * hafxintilawas
  
  c.  haʃxintilawas

- What do we need to project? [+strident]
- What do we need to ban? * [+ant][−ant], * [+ant][−ant]

Example: TSL Samala

```
s

* $ hasxintilawas $  

  ok $ hafxintilawas $ 
```
Unbounded Dependencies are TSL

- Let’s revisit Samala Sibilant Harmony

  (6)  
  a. * ha\textit{s}xintilawaf $
  b. * ha\textit{f}xintilawas $  
  c. ha\textit{f}xintilawaf $

- What do we need to project? [±strident]  
- What do we need to ban? *[±ant][–ant], *[–ant][±ant]

Example: TSL Samala

```
s
.................................
* $ha s x i n t i l a w f $     ok $ha f x i n t i l a w f $
```
Unbounded Dependencies are TSL

Let’s revisit Samala Sibilant Harmony

(6)  a.  * hasxintilawʃ
    b.  * hafxintilawas
    c.  hafʃxintilawaf

What do we need to project? [+strident]
What do we need to ban? *[+ant][−ant], *[−ant][+ant]

Example: TSL Samala

* $hasxintilawaf$  ok $hafxintilawaf$
Unbounded Dependencies are TSL

Let’s revisit Samala Sibilant Harmony

(6)   a.  * hasxintilawais
       b.  * ha∫xintilawas
       c.  ha∫xintilawais

What do we need to project?  [+strident]
What do we need to ban?  * [+ant][−ant], * [−ant][+ant]

Example: TSL Samala

* $hasxintilawais$

$hasxintilawais$

ok $hafxintilawais$
Unbounded Dependencies are TSL

- Let’s revisit Samala Sibilant Harmony

(6)  a. * ha{s}xintilawaʃ
    b. * haʃxintilawas
    c. haʃxintilawaʃ

- What do we need to project? [+strident]
- What do we need to ban? *[++ant][–ant], *[–ant][+ant]

Example: TSL Samala

\[
\begin{array}{c}
s \\
\hline
* \text{has} \text{xintilaw}ʃ \quad \text{ok} \quad \text{has} \text{xintilaw}ʃ
\end{array}
\]
Unbounded Dependencies are TSL

- Let’s revisit Samala Sibilant Harmony
  
  (6)  
  a.  * hasxintilawaf  
  b.  * haʃxintilawas  
  c.  haʃxintilawaf  

- What do we need to project? [+strident]  
- What do we need to ban? *[+ant][−ant],[−ant][+ant]  

Example: TSL Samala

\begin{verbatim}
  s
  .........................
  * $has\\text{xintilawaf}$ $ok$ $ha\\text{f}xintilawaf$
\end{verbatim}
Unbounded Dependencies are TSL

- Let’s revisit Samala Sibilant Harmony

(6) a. * hasxintilawaf
    b. * hafxintilawas
    c. ha]xintilawa[

- What do we need to project? [+strident]
- What do we need to ban? * [+ant][−ant], * [−ant][+ant]

Example: TSL Samala

\[
\begin{array}{c}
\text{s} \\
\text{........................} \\
* \text{hasxintilawaw} & \text{ok hasxintilawaf} \\
\end{array}
\]
Unbounded Dependencies are TSL

- Let’s revisit Samala Sibilant Harmony

(6)  
  a. * hasxintilawaʃ
  b. * haʃxintilawas
  c. haʃxintilawaʃ

- What do we need to project? [+strident]
- What do we need to ban? * [+ant][−ant], * [−ant][+ant]

Example: TSL Samala

```
s
......................
* $hasxintilawaʃ$  ok $haʃxintilawasʃ$
```
Unbounded Dependencies are TSL

Let’s revisit Samala Sibilant Harmony

(6) a. * ha$s$xintilawa$ 
    b. * ha$f$xintilawas 
    c. ha$f$xintilawa$ 

What do we need to project? [+strident]

What do we need to ban? *[+ant][−ant], *[−ant][+ant]

Example: TSL Samala

```
s    f
  ...
* $ha\ s \ x \ i \ n \ t \ i \ l \ a \ w \ f \ $  \ ok \ $ha\ f \ x \ i \ n \ t \ i \ l \ a \ w \ f \ $```

Unbounded Dependencies are TSL

- Let’s revisit Samala Sibilant Harmony

(6) a. * hasxintilawaf
    b. * ha∫xintilawas
    c. ha∫xintilawaf

- What do we need to project? [+strident]
- What do we need to ban? *[+ant][−ant], *[−ant][+ant]

Example: TSL Samala

* $h a s x i n t i l a w f$  $o k$ $h a f x i n t i l a w f$
Unbounded Dependencies are TSL

- Let’s revisit Samala Sibilant Harmony
  
  (6)  
  a.  * hasxintilawaf
  b.  * hafxintilawas
  c.  haʃxintilawaf

- What do we need to project? [+strident]
- What do we need to ban? */+ant][−ant], */−ant][+ant]
  

Example: TSL Samala

```
*sʃ  *sʒ  *zʃ  *zʒ  *ʃs  *ʒs  *ʃz  *ʒz
```

* $ has x i n t i l a w f $ ok $ h a f x i n t i l a w a f $
Unbounded Dependencies are TSL

Let’s revisit Samala Sibilant Harmony

(6)  a. * haˢxintilawaf
    b. * haʃxintilawas
    c. haʃxintilawaf

What do we need to project? [+strident]

What do we need to ban? * [+ant][−ant], * [−ant][+ant]

Example: TSL Samala

* $ha{s \times i n t i l a w f}$

ok $haʃxintilawʃ$
Unbounded Dependencies are TSL

Let’s revisit Samala Sibilant Harmony

(6) a. * hasxintilawaʃ
    b. * haʃxintilawas
    c. haʃxintilawaʃ

What do we need to project? [+strident]
What do we need to ban? * [+ant][−ant], * [−ant][+ant]

Example: TSL Samala

* $\text{hasxintilawaʃ}$
ok $\text{haʃxintilawʃ}$
TSL Phonology: Accounting for Context

- **Unbounded Tone Plateauing in Luganda (UTP)**
  No L may occur within an interval spanned by H.
  (Hyman 2011)

(7)  
- a. LHLLLLL
- b. LLLLLHL
- c. * LHLLHL
- d. LHHHHHL

Example
TSL Phonology: Accounting for Context

- **Unbounded Tone Plateauing in Luganda (UTP)**
  No L may occur within an interval spanned by H. (Hyman 2011)

  (7) a. LLLLLL
  b. LLLLLH
  c. *LHLLLH
  d. LHLLLL

Example
TSL Phonology: Accounting for Context

- **Unbounded Tone Plateauing in Luganda (UTP)**
  No L may occur within an interval spanned by H. (Hyman 2011)

(7)  

a. \text{LHLLLLL}  
b. \text{LLLLLHL}  
c. * \text{LHLLLHL}  
d. \text{LHHHHHL}
TSL Phonology: Accounting for Context

- **Unbounded Tone Plateauing in Luganda (UTP)**
  No L may occur within an interval spanned by H.
  (Hyman 2011)

(7) a. LHLLLLL
    b. LLLLLHL
    c. * LHLLLHL
    d. LHHHHHL

Example

```
L H L L H L

* L H L L H L
```
TSL Phonology: Accounting for Context

- **Unbounded Tone Plateauing in Luganda (UTP)**
  No L may occur within an interval spanned by H.
  (Hyman 2011)

(7)  
  a. LHLLLLL
  b. LLLLLHL
  c. * LHLLHL
  d. LHHHHHL

**Example**

```
L H L L H L
L H L L H L
* L H L L H L
```
Accounting for Context [cont.]

A TSL analysis for UTP (De Santo and Graf 2017):

- Project every H; project L iff immediately follows H
- Ban: HLH

Example

\[ \begin{align*}
\text{ok} & : L H L L L L L \\
* & : L H L L H L
\end{align*} \]

- Most non-local dependencies in phonology are TSL
- What about syntax?
Accounting for Context [cont.]

A TSL analysis for UTP (De Santo and Graf 2017):

- Project every H; project L iff immediately follows H
- Ban: HLH

Example

 Most non-local dependencies in phonology are TSL

What about syntax?
Accounting for Context [cont.]

A TSL analysis for UTP (De Santo and Graf 2017):

- Project every $H$; project $L$ iff immediately follows $H$
- Ban: $HLH$

Example

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>H</th>
<th>L</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ok</td>
<td></td>
<td>L</td>
<td>H</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* $LHLLLHLL$  

- Most non-local dependencies in phonology are TSL
- What about syntax?
Accounting for Context [cont.]

**A TSL analysis for UTP** (De Santo and Graf 2017):

- Project every $H$; project $L$ iff immediately follows $H$
- Ban: $HLH$

**Example**

$H L$

$\ldots\ldots\ldots\ldots\ldots\ldots$

$ok$

$L H L L L L$

$*$ $L H L L L H L$

- Most non-local dependencies in phonology are TSL
- **What about syntax?**
Accounting for Context [cont.]

A TSL analysis for UTP (De Santo and Graf 2017):

- Project every $H$; project $L$ iff immediately follows $H$
- Ban: $HLH$

Example

<table>
<thead>
<tr>
<th>H</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>ok</td>
<td>L</td>
</tr>
</tbody>
</table>

Most non-local dependencies in phonology are TSL

What about syntax?
Accounting for Context [cont.]

A **TSL analysis for UTP** (De Santo and Graf 2017):

- Project every H; project L iff immediately follows H
- Ban: HLH

Example

```
H L
.....................
ok LH L L * LH L L H L
```

- Most non-local dependencies in phonology are TSL
- **What about syntax?**
Accounting for Context [cont.]

**A TSL analysis for UTP** (De Santo and Graf 2017):

- Project every $H$; project $L$ iff immediately follows $H$
- Ban: $HLH$

**Example**

```
  H  L  
  --------
 ok   L H L L L L
```

```
  H  L  
  --------
*    L H L L H L
```

- Most non-local dependencies in phonology are TSL
- **What about syntax?**
Accounting for Context [cont.]

A TSL analysis for UTP (De Santo and Graf 2017):
- Project every $H$; project $L$ iff immediately follows $H$
- Ban: $HLH$

Example

Most non-local dependencies in phonology are TSL
- What about syntax?
Accounting for Context [cont.]

A TSL analysis for UTP (De Santo and Graf 2017):
- Project every $H$; project $L$ iff immediately follows $H$
- Ban: $HLH$

Example

Most non-local dependencies in phonology are TSL
- What about syntax?
Accounting for Context [cont.]

A TSL analysis for UTP (De Santo and Graf 2017):

- Project every $H$; project $L$ iff immediately follows $H$
- Ban: $HLH$

Example

```
ok  L H L L L L
```

```
*  L H L L H L H L
```

- Most non-local dependencies in phonology are TSL
- What about syntax?
Accounting for Context [cont.]

A TSL analysis for UTP (De Santo and Graf 2017):
- Project every $H$; project $L$ iff immediately follows $H$
- Ban: $HLH$

Example

<table>
<thead>
<tr>
<th>$HL$</th>
<th>$HL$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$LHLLL$</td>
<td>$LH[LLH]L$</td>
</tr>
</tbody>
</table>

Most non-local dependencies in phonology are TSL
- What about syntax?
Accounting for Context [cont.]

**A TSL analysis for UTP** (De Santo and Graf 2017):

- Project every **H**; project **L** iff immediately follows **H**
- Ban: **HLH**

**Example**

<table>
<thead>
<tr>
<th><strong>ok</strong></th>
<th><strong>L H L L L L</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>H L</strong></td>
</tr>
</tbody>
</table>

| ***** | **L H L L H L** |

- Most non-local dependencies in phonology are TSL
- **What about syntax?**
Accounting for Context [cont.]

A TSL analysis for UTP (De Santo and Graf 2017):
- Project every H; project L iff immediately follows H
- Ban: HLH

Example

<table>
<thead>
<tr>
<th>ok</th>
<th>L L L L L L</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HL</td>
</tr>
<tr>
<td></td>
<td>HL</td>
</tr>
</tbody>
</table>

Most non-local dependencies in phonology are TSL

What about syntax?
Accounting for Context [cont.]

**A TSL analysis for UTP** (De Santo and Graf 2017):
- Project every $H$; project $L$ iff immediately follows $H$
- Ban: $HLH$

---

**Example**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$HL$</td>
<td>$HLH$</td>
</tr>
</tbody>
</table>

$ok$

| $LHLLL$ | $LHLLHL$ |

$*$

- Most non-local dependencies in phonology are TSL
- What about syntax?
Accounting for Context [cont.]

A TSL analysis for UTP (De Santo and Graf 2017):

- Project every $H$; project $L$ iff immediately follows $H$
- Ban: $HLH$

Most non-local dependencies in phonology are TSL

What about syntax?
Non-Local Dependencies in Syntax

Let’s stick to core operations:

- Move
- Merge?
Non-Local Dependencies in Syntax

Let’s stick to core operations:

- Move
- Merge?

```
Merge
  Mary
    D^-
  Merge
    's
      N^+
      D^+
      D^-
    cat
      N^-
```
Non-Local Dependencies in Syntax

Let’s stick to core operations:
- Move
- **Merge**: Unbounded adjunction

Frey and Gärtner (2002); Graf (2017b)
Merge with Adjunction is TSL

A TSL grammar for Merge
Merge with Adjunction is TSL

A TSL grammar for Merge

1 Project Merge iff a child has $X^+$ (e.g. $X = N$)
Merge with Adjunction is TSL

A TSL grammar for Merge

1. Project **Merge** iff a child has $X^+$ (e.g. $X = N$)
Merge with Adjunction is TSL

A TSL grammar for Merge

1. Project Merge iff a child has $X^+$ (e.g. $X = N$)
2. Project any node which has $X^-$ (e.g. $X = N$)
Merge with Adjunction is TSL

A TSL grammar for Merge

1. Project **Merge** iff a child has $X^+$ (e.g. $X = N$)
2. Project any node which has $X^-$ (e.g. $X = N$)
Merge with Adjunction is TSL

A TSL grammar for Merge

1. Project **Merge** iff a child has \(X^+\) (e.g. \(X = N\))
2. Project any node which has \(X^-\) (e.g. \(X = N\))
Merge with Adjunction is TSL

A TSL grammar for Merge

1. Project **Merge** iff a child has $X^+$ (e.g. $X = N$)
2. Project any node which has $X^-$ (e.g. $X = N$)
Merge with Adjunction is TSL

A TSL grammar for Merge

1. Project **Merge** iff a child has $X^+$ (e.g. $X = N$)
2. Project any node which has $X^-$ (e.g. $X = N$)
Merge with Adjunction is TSL

A TSL grammar for Merge

1. Project $\textbf{Merge}$ iff a child has $X^+$ (e.g. $X = N$)
2. Project any node which has $X^-$ (e.g. $X = N$)
Merge with Adjunction is TSL

A TSL grammar for Merge

1. Project Merge iff a child has $X^+$ (e.g. $X = N$)
2. Project any node which has $X^-$ (e.g. $X = N$)
3. No Merge without exactly one LI among its daughters.
Merge with Adjunction is TSL

A TSL grammar for Merge

1. Project **Merge** iff a child has $X^+$ (e.g. $X = V$)
2. Project any node which has $X^-$ (e.g. $X = V$)
3. No Merge without exactly one LI among its daughters.
Merge with Adjunction is TSL

A TSL grammar for Merge

1. Project Merge iff a child has $X^+$ (e.g. $X = V$)
2. Project any node which has $X^-$ (e.g. $X = V$)
3. No Merge without exactly one LI among its daughters.
Merge with Adjunction is TSL

A TSL grammar for Merge

1. Project **Merge** iff a child has \( X^- \) (e.g. \( X = V \))
2. Project any node which has \( X^+ \) (e.g. \( X = V \))
3. No Merge without exactly one LI among its daughters.
Merge with Adjunction is TSL

A TSL grammar for Merge

1. Project **Merge** iff a child has $X^{-}$ (e.g. $X = V$)
2. Project any node which has $X^{+}$ (e.g. $X = V$)
3. No Merge without exactly one LI among its daughters.
Merge with Adjunction is TSL

A TSL grammar for Merge

1. Project Merge iff a child has $X^-$ (e.g. $X = V$)
2. Project any node which has $X^+$ (e.g. $X = V$)
3. No Merge without exactly one LI among its daughters.
TSL Merge: Multiple Tiers

![Diagram of TSL Merge: Multiple Tiers]
TSL Merge: Multiple Tiers

Merge

Mary

D−

Merge

Merge

Adjoin

’s

N+ D+ D−

Adjoin

stinky

cat

old

N−

N−
TSL Merge: Multiple Tiers
TSL Merge: Understanding the Constraint

*Merge

the Merge

N⁺ D⁻

Mary Merge

D⁻

's Adjoin

N⁺ D⁺ D⁻

stinky Adjoin

old cat

N⁻
TSL Merge: Understanding the Constraint

*Merge

the *Merge

Mary Merge

D−

's Adjoin

N+ D+ D−

stinky Adjoin

old cat

N−
TSL Merge: Understanding the Constraint

*Merge

the Merge

N⁺ D⁻

Mary Merge

D⁻

's Adjoin

N⁺ D⁺ D⁻

stinky Adjoin

old cat

N⁻
TSL Merge: Understanding the Constraint
TSL Merge: Understanding the Constraint

*Merge

the Merge

Mary Merge
d_merge

's Merge

stinky Adjoin

old cat

N^+ D^- N^+ D^+ D^- D^-
TSL Merge: Understanding the Constraint

*Merge

- Merge
  - the
    - N^+
    - D^-
  - Mary
    - D^-
  - Merge
  - Merge
    - Merge
      - 's
        - N^+
        - D^+
        - D^-
      - stinky
      - Adjoin
        - old
        - cat
        - N^-
Parallels Between Phonology And Syntax

<table>
<thead>
<tr>
<th></th>
<th>Local</th>
<th>Non-local</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phonology</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Syntax</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

- Relativized Locality:
  Non-local dependencies are local over a simple relativization domain.

Strong Cognitive Parallelism Hypothesis

Phonology, (morphology), and syntax have the same subregular complexity over their respective structural representations.
Parallels Between Phonology And Syntax

<table>
<thead>
<tr>
<th></th>
<th>Local</th>
<th>Non-local</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phonology</td>
<td>SL</td>
<td>?</td>
</tr>
<tr>
<td>Syntax</td>
<td>SL</td>
<td>?</td>
</tr>
</tbody>
</table>

Relativized Locality:
Non-local dependencies are local over a simple relativization domain.

Strong Cognitive Parallelism Hypothesis
Phonology, (morphology), and syntax have the same subregular complexity over their respective structural representations.
Parallels Between Phonology And Syntax

<table>
<thead>
<tr>
<th></th>
<th>Local</th>
<th>Non-local</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Phonology</strong></td>
<td>SL</td>
<td>TSL</td>
</tr>
<tr>
<td><strong>Syntax</strong></td>
<td>SL</td>
<td>TSL</td>
</tr>
</tbody>
</table>

- **Relativized Locality:**
  Non-local dependencies are local over a simple relativization domain.

**Strong Cognitive Parallelism Hypothesis**

Phonology, (morphology), and syntax have the same subregular complexity over their respective structural representations.
## Parallels Between Phonology And Syntax

<table>
<thead>
<tr>
<th></th>
<th>Local</th>
<th>Non-local</th>
<th>Data Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Phonology</strong></td>
<td>SL</td>
<td>TSL</td>
<td>Strings</td>
</tr>
<tr>
<td><strong>Syntax</strong></td>
<td>SL</td>
<td>TSL</td>
<td>Trees</td>
</tr>
</tbody>
</table>

- **Relativized Locality:**
  Non-local dependencies are local over a simple relativization domain.

---

### Strong Cognitive Parallelism Hypothesis

Phonology, (morphology), and syntax have the same subregular complexity over their respective structural representations.
Parallels Between Phonology And Syntax

<table>
<thead>
<tr>
<th></th>
<th>Local</th>
<th>Non-local</th>
<th>Data Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phonology</td>
<td>SL</td>
<td>TSL</td>
<td>Strings</td>
</tr>
<tr>
<td>Syntax</td>
<td>SL</td>
<td>TSL</td>
<td>Trees</td>
</tr>
</tbody>
</table>

- **Relativized Locality:**
  Non-local dependencies are local over a simple relativization domain.

**Strong Cognitive Parallelism Hypothesis**

Phonology, (morphology), and syntax have the same subregular complexity over their respective structural representations.
A Bird’s-Eye View of the Framework

- recursively enumerable
- context-sensitive
- mildly-context sensitive
- context-free
- regular
- TSL
A Bird’s-Eye View of the Framework

- **Phonology**
  - Kaplan and Kay (1994)
  - Strings

- **Syntax**
  - Shieber (1985)
  - Strings

- **Morphology**
  - Karttunen et al. (1992)
  - Strings

- **TSL**
  - Regular
  - Context-free
  - Mildly-context sensitive
  - Context-sensitive
  - Recursively enumerable

Kaplan and Kay (1994) and Shieber (1985) provide insights into the linguistic structures, while Karttunen et al. (1992) offer a different perspective.
A Bird’s-Eye View of the Framework
Conclusion

Strong Cognitive Parallelism Hypothesis

Phonology, (morphology), and syntax have the same subregular complexity over their respective structural representations.

We gain a unified perspective on:

- typology
- learnability
- cognition
Conclusion

**Strong Cognitive Parallelism Hypothesis**

Phonology, (morphology), and syntax have the same subregular complexity over their respective structural representations.

**We gain a unified perspective on:**

- **typology**
  - Intervocalic Voicing iff applied an even times in the string
  - Have a CP iff it dominates ≥ 3 TPs

- **learnability**

- **cognition**
Conclusion

**Strong Cognitive Parallelism Hypothesis**

Phonology, (morphology), and syntax have the same subregular complexity over their respective structural representations.

**We gain a unified perspective on:**

- **typology**
  - Intervocalic Voicing iff applied *an even times* in the string
  - Have a CP iff it dominates $\geq 3$ TPs

- **learnability**
  Learnable from positive examples of strings/trees.

- **cognition**
Conclusion

Strong Cognitive Parallelism Hypothesis

Phonology, (morphology), and syntax have the same subregular complexity over their respective structural representations.

We gain a unified perspective on:

- typology
  - Intervocalic Voicing iff applied an even times in the string
  - Have a CP iff it dominates ≥ 3 TPs

- learnability
  Learnable from positive examples of strings/trees.

- cognition
  Finite, flat memory
Future Work

We are just getting started:

- autosegmental structures (Jardine 2017:i.a)
- morphological derivations (Chandlee 2017; Aksënova and De Santo 2017)
- mappings (Chandlee 2014; Chandlee and Heinz 2018:i.a.)
- syntax beyond Merge and Move (Graf 2017b; Vu 2018)

Join the Enterprise!

- typological universals/gaps
- TSL-analyses of phenomena/counterexamples
- artificial language learning experiments
- new formal results
- and much more ...


Work by Alëna Aksënova, Thomas Graf, and Sophie Moradi.

It seems that **morphology is also TSL**. (Aksënova et al. 2016)

Morphology ≡ Morphotactics of underlying forms but see (Aksënova and De Santo 2017) on derivations

We are unaware of any non-TSL patterns in this realm.

Tight typology, explains gaps
Example: Circumfixation in Indonesian

- Indonesian has circumfixation with no upper bound on the distance between the two parts of the circumfix.

\[(8) \text{ maha siswa} \quad \text{big pupil} \quad \text{‘student’} \]
\[(9) *(ke-) \text{ maha siswa } *(-an) \quad \text{NMN- big pupil -NMN} \quad \text{‘student affairs’} \]

- Requirements: exactly one $ke$- and exactly one $-an$

| Tier$^1$ | contains all $\text{NMN}$ affixes | \$ an \$ ke ke \$ |
| Tier$^0$ | contains all morphemes | \$an, ke\$, keke, anan \$ |
| \text{n-grams} | \$an, ke\$, keke, anan \$ |
Example: Circumfixation in Indonesian

- Indonesian has circumfixation with no upper bound on the distance between the two parts of the circumfix.

\[(8) \text{ maha siswa } \quad \text{big pupil 'student'}\]

\[(9) *(\text{ke-}) \text{ maha siswa } *(-\text{an}) \quad \text{NMN- big pupil -NMN 'student affairs'}\]

- Requirements: exactly one \textit{ke-} and exactly one \textit{-an}

\textbf{Tier}_1 \quad \text{contains all \textit{NMN} affixes} \quad $\text{an} \quad \text{ke} \quad \text{ke} \quad $\n
\textbf{Tier}_0 \quad \text{contains all morphemes} \quad | \quad | \quad | \quad | \quad | \quad | \quad |

\textit{n-grams} \quad $\textit{an, ke}$, \textit{keke}, \textit{anan} \quad $\text{an} \quad \text{m} \quad \text{s} \quad \text{ke} \quad \text{ke} \quad $
Swahili *vyo* is **either a prefix or a suffix**, depending on presence of negation. (?)

(10)  a.  a- vi- **soma** -vyo  
SBJ:CL.1- OBJ:CL.8- read -REL:CL.8  
‘reads’

b.  a- si- **vyo**- vi- **soma**  
SBJ:CL.1- NEG- REL:CL.8- read -OBJ:CL.8  
‘doesn’t read’
Example: Swahili *vyo [cont.]

(11) a. *a- **vyo-** vi- **soma**
   SBJ:CL.1- REL:CL.8- OBJ:CL.8- read

b. *a- **vyo-** vi- **soma** -vyo
   SBJ:CL.1- REL:CL.8- OBJ:CL.8- read -REL:CL.8

c. *a- **si-** **vyo-** vi- **soma**
   SBJ:CL.1- NEG- REL:CL.8- OBJ:CL.8- read
   -vyo
   REL:CL.8-

d. *a- **si-** vi- **soma** -vyo
   SBJ:CL.1- NEG- OBJ:CL.8- read REL:CL.8-
Example: Swahili *vyo* [cont.]

### Generalizations About *vyo*

- may occur at most once
- must follow negation prefix *si*- if present
- is a prefix iff *si*- is present

<table>
<thead>
<tr>
<th>Tier&lt;sub&gt;1&lt;/sub&gt;</th>
<th>contains <em>vyo</em>, <em>si</em>, and stem edges #</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tier&lt;sub&gt;0&lt;/sub&gt;</strong></td>
<td>contains all morphemes</td>
</tr>
<tr>
<td><em>n</em>-grams</td>
<td><em>vyovyoyo, vyo##vyo</em> “at most one <em>vyo</em>”</td>
</tr>
<tr>
<td></td>
<td><em>vyosi, vyo##si</em> “<em>vyo</em> follows <em>si</em>”</td>
</tr>
<tr>
<td></td>
<td><em>si##vyo, $vyo$##</em> “<em>vyo</em> is prefix iff <em>si</em> present”</td>
</tr>
</tbody>
</table>
Restriction to TSL can also explain some typological gaps.

**General Strategy**

- Attested patterns A and B are TSL.
- But combined pattern A+B is not attested.
- Show that A+B is not TSL.
Example: Compounding Markers

- Russian has an infix -o- that may occur between parts of compounds.
- Turkish has a single suffix -si that occurs at end of compounds.

(12) vod -o- voz -o- voz
    water -COMP- carry -COMP- carry
    ‘carrier of water-carriers’

(13) türk bahçe kapı -si (/*-si)
    turkish garden gate -COMP (/*-COMP)
    ‘Turkish garden gate’

- **New Universal**
  If a language allows unboundedly many compound affixes, they are **infixes**.
Example: Compounding Markers [cont.]

- Russian and Turkish are TSL.

  $\text{Tier}_1$ \textsc{comp} affix and stem edges $\#$

  - Russian $n$-grams: $\text{o}_0$, $\text{o}_1$, $\text{o}_2$
  - Turkish $n$-grams: $\text{sisi}$, $\text{si}_1$, $\text{si}_2$

- The combined pattern would yield Ruskish: stem$^{n+1}$-si$^m$

- This pattern is not regular and hence not TSL either.
Interim Summary: Morphology

▶ While we know less about morphology than phonology at this point, it also seems to be TSL.
▶ Even complex patterns like Swahili *vyo* can be captured.
▶ At the same time, we get **new universals**:

**Bounded Circumfixation**  No recursive process can be realized via circumfixation.

▶ We can reuse tools and techniques from TSL phonology, including learning algorithms.
▶ The cognitive resource requirements are also comparable.
**MGs & Derivation Trees**

Phrase Structure Tree
MGs & Derivation Trees

Phrase Structure Tree

Derivation Tree
## Constraints on Move

### What about Move?

Suppose our MG is in **single movement normal form**, i.e. every phrase moves at most once. Then movement is regulated by two constraints. (Graf 2012a)

<table>
<thead>
<tr>
<th>Constraints on Movement</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Move</strong>  Every head with a negative Move feature is dominated by a matching Move node.</td>
</tr>
<tr>
<td><strong>SMC</strong> Every Move node is a closest dominating match for exactly one head.</td>
</tr>
</tbody>
</table>
What about Move?

Suppose our MG is in **single movement normal form**, i.e. every phrase moves at most once.

Then movement is regulated by two constraints. (Graf 2012a)

<table>
<thead>
<tr>
<th>Constraints on Movement</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Move</strong></td>
</tr>
<tr>
<td><strong>SMC</strong></td>
</tr>
</tbody>
</table>
Tiers for Movement

- There is no upper bound on the distance between a lexical item and its matching Move node.
- Consequently, **Move dependencies are not local**.
- What if every movement type (wh, topic, . . . ) induces its own tier? Would that make Move dependencies local?
Tiers for Movement

- There is no upper bound on the distance between a lexical item and its matching Move node.
- Consequently, **Move dependencies are not local**.
- What if every movement type (wh, topic, ...) induces its own tier? Would that make Move dependencies local?
Tiers for Movement

- There is no upper bound on the distance between a lexical item and its matching Move node.
- Consequently, **Move dependencies are not local.**
- What if every movement type (wh, topic, ...) induces its own tier? Would that make Move dependencies local?

```
Move
  |   Merge
  |     Move
  |       Merge
  |         Move
  |           Merge
  |               Move
  a   b   c   d   e
```

```
Move
  |   Merge
  |     Merge
  f
  e
```
Tiers for Movement

- There is no upper bound on the distance between a lexical item and its matching Move node.
- Consequently, **Move dependencies are not local**.
- What if every movement type (wh, topic, ...) induces its own tier? Would that make Move dependencies local?
Move Constraints over Tiers

**Original**

- **Move**: Every head with a negative Move feature is dominated by a matching Move node.
- **SMC**: Every Move node is a closest dominating match for exactly one head.

**Tier**

- Every lexical item has a **mother** labeled Move.
- Exactly one of a Move node’s **daughters** is a lexical item.

**Tree n-gram Templates**

<table>
<thead>
<tr>
<th></th>
<th>Move</th>
<th>SMC1</th>
<th>SMC2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move</td>
<td>$</td>
<td>Move</td>
<td>Move</td>
</tr>
<tr>
<td></td>
<td>≥ 1 LI</td>
<td>no LI</td>
<td>≥ 2 LIs</td>
</tr>
</tbody>
</table>
Example of Ill-Formed Derivation
Example of Ill-Formed Derivation

References
Example of Ill-Formed Derivation

$\begin{array}{c}
\text{Merge} \\
g \quad \text{Move} \\
\quad \text{Merge} \\
\quad \text{Merge} \\
\text{Merge} \quad c \quad \text{Move} \quad f \\
\quad a \quad b \quad \text{Merge} \\
\quad \quad \quad d \quad e \\
\end{array}$

SMC1

SMC2

$\geq 1 \text{ LI}$

no LI

$\geq 2 \text{ LI}$
Example of Ill-Formed Derivation
Example of Ill-Formed Derivation
Example of Ill-Formed Derivation

References

\[ \text{Merge} \]
\[ g \quad \text{Move} \]
\[ \quad \text{Merge} \]
\[ \quad \text{Merge} \]
\[ \text{Merge} \quad c \quad \text{Move} \quad f \]
\[ a \quad b \]
\[ \text{Merge} \]
\[ d \quad e \]

\[ \geq 1 \text{ LI} \]
\[ \text{Move} \]
\[ \geq 2 \text{ LI} \]
\[ \text{SMC2} \]
\[ \geq 1 \text{ LI} \]

\[ \text{Move} \]
\[ \text{no LI} \]
\[ \text{SMC1} \]
Example of Well-Formed Derivation
Example of Well-Formed Derivation
Example of Well-Formed Derivation

```
Move
  |
Merge
  |
Move  Merge
  |
Move  f
  |
Merge
  |
c
  |
Merge
  |
da
d  e

$  

$  

Move  f
  |
a
  |
$  

$  

Move
  |
e
  |
$  

$  

≥ 1 LI
Move

no LI
SMC1

≥ 2 LI
SMC2
```
Remarks on Single Movement Normal Form

- Single Movement Normal Form seems unrealistic.
- **But:** does not rule out multiple movement steps, only says there is **single feature trigger in derivation**
- Intermediate landing sites can be part of structure built from the derivation tree.

A Conjecture on Movement Restrictions (Graf 2017a)

- Conversion of an MG into single movement normal form causes large blow-up in size of lexicon.
- Blow-up varies a lot: from 0 to hundred times the original size
- The more fixed the position of movers, the smaller the blow-up
  ⇒ island constraints as a means to limit lexical blow-up?
Remarks on Single Movement Normal Form

- Single Movement Normal Form seems unrealistic.
- **But:** does not rule out multiple movement steps, only says there is *single feature trigger in derivation*
- Intermediate landing sites can be part of structure built from the derivation tree.

A Conjecture on Movement Restrictions (Graf 2017a)

- Conversion of an MG into single movement normal form causes large blow-up in size of lexicon.
- Blow-up varies a lot: from 0 to hundred times the original size
- The more fixed the position of movers, the smaller the blow-up
  \[ \Rightarrow \text{island constraints as a means to limit lexical blow-up?} \]
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
  ⇒ central data structure of Minimalist syntax
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. 2012)
  - crossing < nested dependencies (Kobele et al. 2012)
  - SC-RC < RC-SC (?)
  - SRC < ORC in English (?)
  - SRC < ORC in East-Asian (?)
  - quantifier scope preferences (Pasternak 2016)
Technical Fertility of Derivation Trees

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

- sideways 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)
Sibilant Harmony in **Samala** (McMullin 2016)

1) Unbounded sibilant harmony

<table>
<thead>
<tr>
<th>Word Structure</th>
<th>Resultant</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>/k-su-[jojin/</code></td>
<td>kʃuʃojin</td>
<td>“I darken it”</td>
</tr>
<tr>
<td><code>/k-su-k’ili-mekteken-ʃ/</code></td>
<td>kʃuk’ilimekeketʃ</td>
<td>“I straighten up”</td>
</tr>
</tbody>
</table>

2) `/s/ → [ʃ]` when preceding (adjacent) `[t, n, l]`

<table>
<thead>
<tr>
<th>Word Structure</th>
<th>Resultant</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>/s-lok’in/</code></td>
<td>ʃlok’in</td>
<td>“he cuts it”</td>
</tr>
<tr>
<td><code>/s-tepu?/</code></td>
<td>ʃtepu?</td>
<td>“he gambles”</td>
</tr>
</tbody>
</table>

3) Long-distance agreement overrides local disagreement

<table>
<thead>
<tr>
<th>Word Structure</th>
<th>Resultant</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>/s-iʃt-iʃti-jep-us/</code></td>
<td>sististijepus</td>
<td>“they show him”</td>
</tr>
<tr>
<td><code>/s-net-us/</code></td>
<td>snetus</td>
<td>“he does it to him”</td>
</tr>
</tbody>
</table>
Sibilant Harmony in **Samala** (McMullin 2016)

1) Unbounded sibilant harmony
   
   a. /k-su-ʃojin/  kʃuʃojin  "I darken it"
   b. /k-su-k’ili-mekeken-ʃ/  kʃuk’ilimekeketʃ  "I straighten up"

2) /s/ → [ʃ] when preceding (adjacent) [t, n, l]
   
   a. /s-lok’in/  šlok’in  "he cuts it"
   b. /s-tepuʔ/  štepuʔ  "he gambles"

3) Long-distance agreement overrides local disagreement
   
   a. /s-ʃt-ʃti-jep-us/  šstitstijepus  "they show him"
   b. /s-net-us/  šnetus  "he does it to him"
# Samala (Revisited)

## Sibilant Harmony in Samala (McMullin 2016)

1) Unbounded sibilant harmony

<table>
<thead>
<tr>
<th>/k-su-	ext{j}ojin/</th>
<th>kʃuʃojin</th>
<th>“I darken it”</th>
</tr>
</thead>
<tbody>
<tr>
<td>a. /k-su-	ext{j}ojin/</td>
<td>kʃuʃojin</td>
<td>“I darken it”</td>
</tr>
<tr>
<td>b. /k-su-k’ili-mekteken-	ext{j}/</td>
<td>kʃuk’ilimekeketʃ</td>
<td>“I straighten up”</td>
</tr>
</tbody>
</table>

2) /s/ → [ʃ] when preceding (adjacent) [t, n, l]

<table>
<thead>
<tr>
<th>/s-lok’in/</th>
<th>ʃlok’in</th>
<th>“he cuts it”</th>
</tr>
</thead>
<tbody>
<tr>
<td>a. /s-lok’in/</td>
<td>ʃlok’in</td>
<td>“he cuts it”</td>
</tr>
<tr>
<td>b. /s-tepu?/</td>
<td>ʃtepu?</td>
<td>“he gambles”</td>
</tr>
</tbody>
</table>

3) Long-distance agreement overrides local disagreement

<table>
<thead>
<tr>
<th>/s-ʃt-ʃti-jep-us/</th>
<th>sististijepus</th>
<th>“they show him”</th>
</tr>
</thead>
<tbody>
<tr>
<td>a. /s-ʃt-ʃti-jep-us/</td>
<td>sististijepus</td>
<td>“they show him”</td>
</tr>
<tr>
<td>b. /s-net-us/</td>
<td>snetus</td>
<td>“he does it to him”</td>
</tr>
</tbody>
</table>
Structure-Sensitive TSL (SS-TSL)

SAMALA Sibilant Harmony (Revisited)

- anticipatory sibilant harmony
- palatalization to avoid local restrictions
- sibilant harmony overrides palatalization
Structure-Sensitive TSL (SS-TSL)

**Samala Sibilant Harmony (Revisited)**

- anticipatory sibilant harmony
- palatalization to avoid local restrictions
- sibilant harmony overrides palatalization

\[ T = \{ \sigma : \sigma \in \{ s, S \} \lor (\sigma \in \{ n, t, l \} \land s \prec + \sigma) \} \]
\[ S = \{ *sS, *sS, *sn(\neg s), *st(\neg s), *sl(\neg s) \} \]
Structure-Sensitive TSL (SS-TSL)

SAMALA Sibilant Harmony (Revisited)

- anticipatory sibilant harmony
- palatalization to avoid local restrictions
- sibilant harmony overrides palatalization
Structure-Sensitive TSL (SS-TSL)

**SAMALA Sibilant Harmony (Revisited)**

- anticipatory sibilant harmony
- palatalization to avoid local restrictions
- sibilant harmony overrides palatalization

\[
\begin{align*}
T &= \{ \sigma : \sigma \in \{ s, S \} \lor (\sigma \in \{ n, t, l \} \land s \prec \sigma) \} \\
S &= \{ \ast s S, \ast s S, \ast sn(\neg s), \ast st(\neg s), \ast sl(\neg s) \}
\end{align*}
\]
Structure-Sensitive TSL (SS-TSL)

**Samala Sibilant Harmony (Revisited)**

- anticipatory sibilant harmony
- palatalization to avoid local restrictions
- sibilant harmony overrides palatalization

```
s n

\[ \times s n e t u s \times \]```
Structure-Sensitive TSL (SS-TSL)

**SAMALA Sibilant Harmony (Revisited)**

- anticipatory sibilant harmony
- palatalization to avoid local restrictions
- sibilant harmony overrides palatalization

\[
\sigma, \sigma' \in \{s, S\} \lor \sigma \in \{n, t, l\} \land s \prec \sigma\]

\[
\ast s S, \ast s S, \ast s n (\neg s), \ast s t (\neg s), \ast s l (\neg s)
\]

\[\times s n e t u s \times\]
Structure-Sensitive TSL (SS-TSL)

**Samala Sibilant Harmony (Revisited)**

- anticipatory sibilant harmony
- palatalization to avoid local restrictions
- sibilant harmony overrides palatalization

```
s n
```

```
⊗ s n e t u s ⊗
```
Structure-Sensitive TSL (SS-TSL)

**SAMALA Sibilant Harmony (Revisited)**

- anticipatory sibilant harmony
- palatalization to avoid local restrictions
- sibilant harmony overrides palatalization

\[
\begin{align*}
S & = \{ *sS, *sS, *sn(\neg s), *st(\neg s), *sl(\neg s) \} \\
T & = \{ \sigma : \sigma \in \{ s, S \} \lor (\sigma \in \{ n, t, l \} \land s \prec + \sigma) \} 
\end{align*}
\]
Structure-Sensitive TSL (SS-TSL)

Samala Sibilant Harmony (Revisited)

- anticipatory sibilant harmony
- palatalization to avoid local restrictions
- sibilant harmony overrides palatalization
Structure-Sensitive TSL (SS-TSL)

**SAMALA** Sibilant Harmony (Revisited)

- anticipatory sibilant harmony
- palatalization to avoid local restrictions
- sibilant harmony overrides palatalization

\[ T = \{ \sigma : \sigma \in \{s, S\} \lor (\sigma \in \{n, t, l\} \land s \prec +\sigma) \} \]

\[ S = \{ *ss, *ss, *sn(\neg s), *st(\neg s), *sl(\neg s) \} \]

\[ *\]

\[ s \text{ } n \text{ } s \]

\[ \times \text{ } s \text{ } n \text{ } e \text{ } t \text{ } u \text{ } s \text{ } \times \]
Structure-Sensitive TSL (SS-TSL)

**SAMALA Sibilant Harmony (Revisited)**

- anticipatory sibilant harmony
- palatalization to avoid local restrictions
- sibilant harmony overrides palatalization

\[
\begin{align*}
T = \{ &\sigma : \sigma \in \{s, S\} \\
&\lor (\sigma \in \{n, t, l\} \land s \prec + \sigma) \}
\end{align*}
\]

\[
S = \{ \ast ssS, \ast ssS, \ast sn(\neg s), \ast st(\neg s), \ast sl(\neg s) \}
\]
Structure-Sensitive TSL (SS-TSL)

**SAMALA** Sibilant Harmony (Revisited)

- anticipatory sibilant harmony
- palatalization to avoid local restrictions
- sibilant harmony overrides palatalization

Grammar

\[
T = \{ \sigma : \sigma \in \{s, S\} \lor (\sigma \in \{n, t, l\} \land s \prec \sigma) \} \\
S = \{(*sS), (*sS), (*sn(\neg s)), (*st(\neg s)), (*sl(\neg s)) \}
\]

The diagram illustrates a possible sequence in Samala, showing sibilant harmony and its interaction with palatalization. The sequence is marked as "ok" and includes a specific order of pronunciation, indicating that the sibilant harmony is preserved even when palatalization is applied.
Structure-Sensitive TSL (SS-TSL)

**Samala Sibilant Harmony (Revisited)**

- anticipatory sibilant harmony
- palatalization to avoid local restrictions
- sibilant harmony overrides palatalization

**Grammar**

\[
T = \{ \sigma : \sigma \in \{ s, \emptyset \} \lor (\sigma \in \{ n, t, l \} \land s \prec^+ \sigma) \} \\
S = \{ *s\emptyset, *s\emptyset, *sn(\neg s), *st(\neg s), *sl(\neg s) \}
\]
SS -TSL: Relations to other Classes
The TSL Neighborhood: a Plethora of Combinations

- Regular
  - SF
    - $\cap$-TESL
      - $\cap$-SS-TSL
        - MTSL
        - SS-TSL
      - TESL
      - LTT
    - TSL
    - LT
    - PT
  - MTSL
  - SP