Mathematical Linguistics & Cognitive Complexity

Aniello De Santo

aniellodesanto.github.io
aniello.desanto@utah.edu
@AnyDs

NTNU
May 24, 2022

(Some) Big Questions

- Are there **laws** that govern linguistic knowledge?
- **Why** are those the laws?
- Do they relate **typological gaps**?
- (How) are the reflected in **human cognitive processes**?
- What can we infer about **linguistic representations**?

Cross-disciplinarity for the win

- Stand on the shoulders of giants.
- Cross-fertilization and multiple explanatory levels.
- Yields new generalizations and data.

(Some) Big Questions

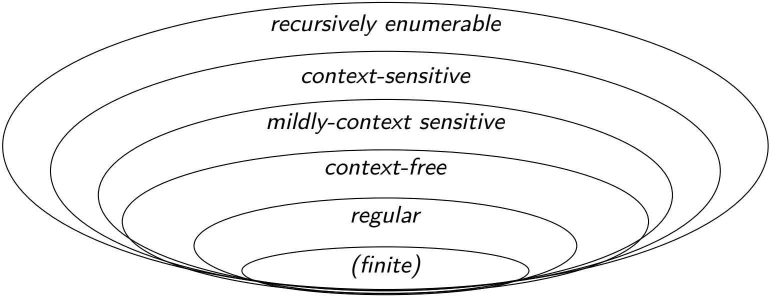
- Are there **laws** that govern linguistic knowledge?
- **Why** are those the laws?
- Do they relate **typological gaps**?
- (How) are the reflected in **human cognitive processes**?
- What can we infer about **linguistic representations**?

Cross-disciplinarity for the win

- Stand on the shoulders of giants.
- Cross-fertilization and multiple explanatory levels.
- Yields new generalizations and data.

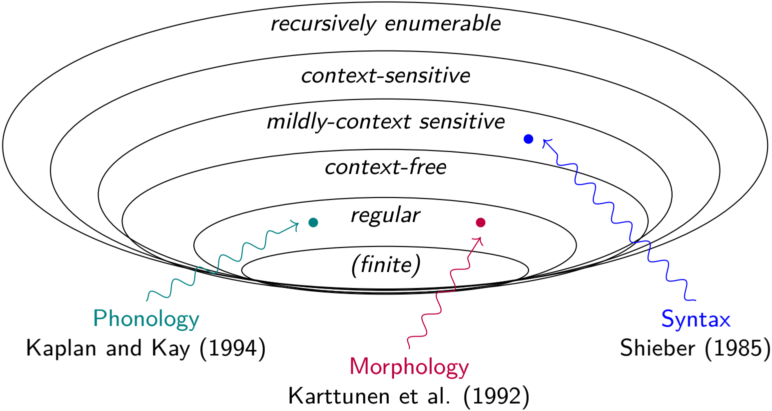
Computational Theories of Language

Languages (stringsets) can be classified according to the complexity of the grammars that generate them.

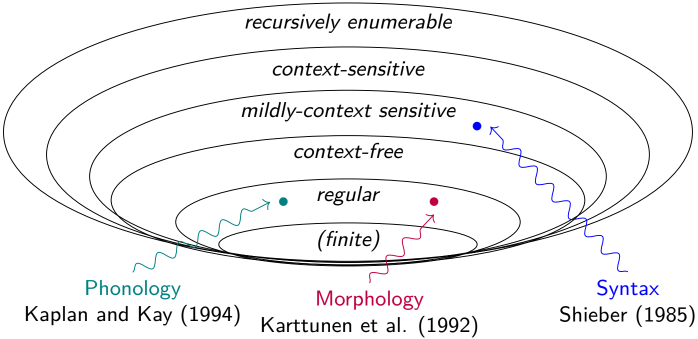


Computational Theories of Language

Languages (stringsets) can be classified according to the complexity of the grammars that generate them.



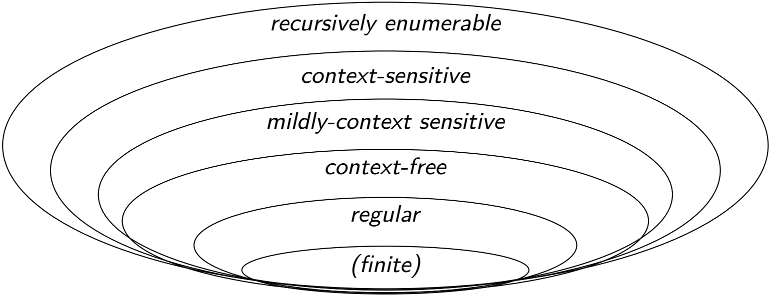
Precise Theories ⇒ Precise Predictions



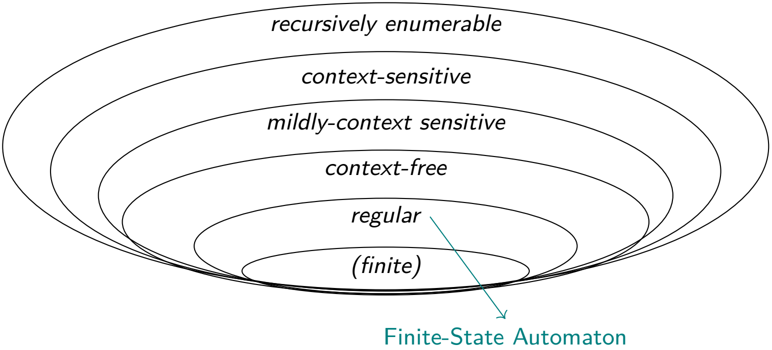
Precise predictions for:

- typology → e.g. no center embedding in phonology
- learnability → e.g. no Gold learning for regular languages
- cognition?

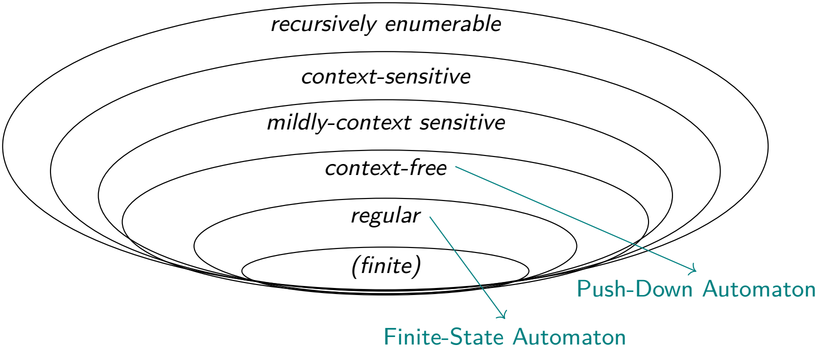
Chomsky Hierarchy and Automata Theory



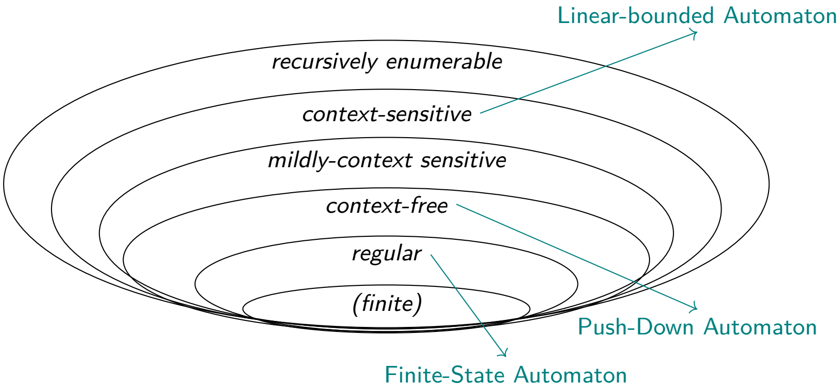
Chomsky Hierarchy and Automata Theory



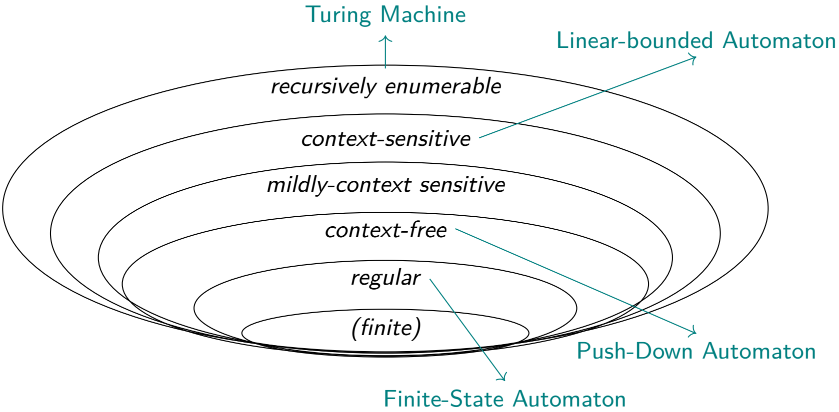
Chomsky Hierarchy and Automata Theory



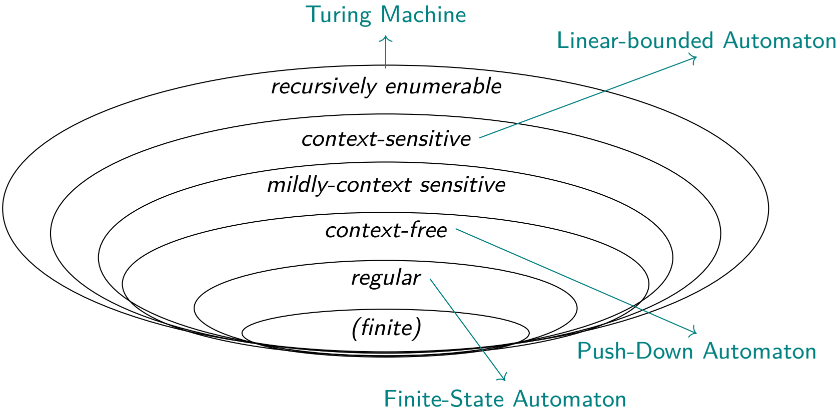
Chomsky Hierarchy and Automata Theory



Chomsky Hierarchy and Automata Theory



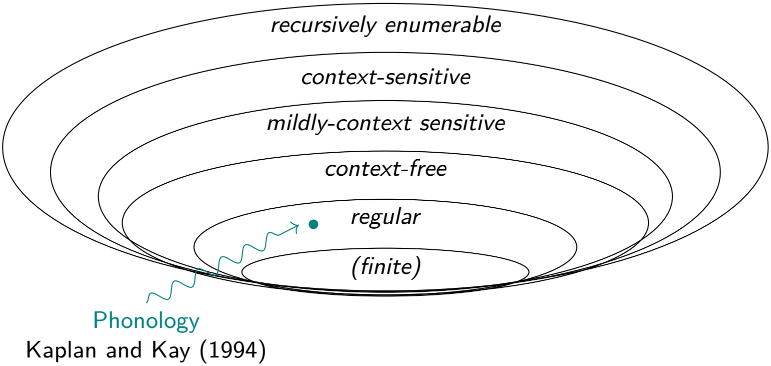
Chomsky Hierarchy and Automata Theory



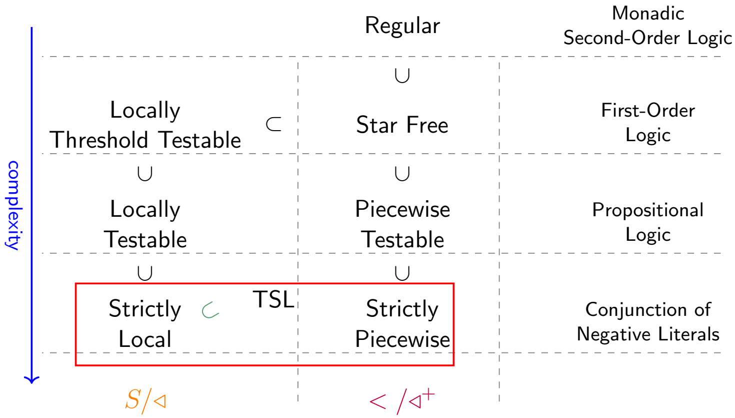
Automata theoretic classes seem to presuppose [...] specific classes of recognition mechanisms, raising questions about whether these are necessarily relevant to the cognitive mechanisms under study.

Rogers & Pullum 2011

Phonology as a Regular System

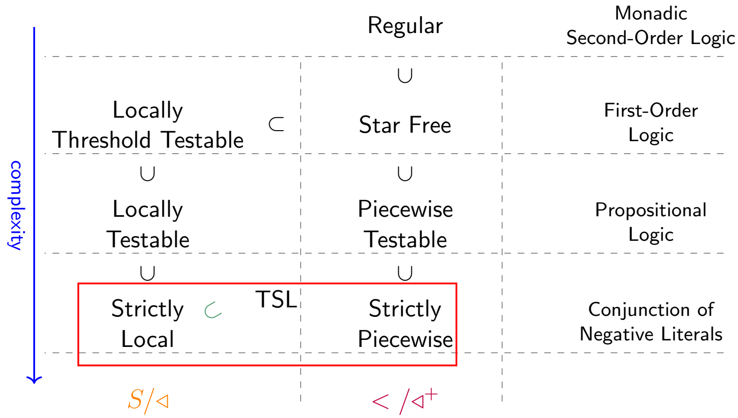


Beyond Monolithic Classes: Subregular Languages



- Multiple equivalent characterizations: algebraic, logic, automata...

Beyond Monolithic Classes: Subregular Languages



- Multiple equivalent characterizations: algebraic, logic, automata...

Outline

Some Insights

Parallels between phonology and syntax?

- What would a computational linguist tell you?
  Well, it depends!
- What will I show you?
  They are fundamentally similar!

The Take-Home Message

- **Two kind of dependencies**: local and non-local
- The core mechanisms are **the same** cross-domain
- That is: linguistic dependencies are **local** over the right **structural representations**

Some Insights

Parallels between phonology and syntax?

- What would a computational linguist tell you?
  Well, it depends!
- What will I show you?
  They are fundamentally similar!

The Take-Home Message

- **Two kind of dependencies**: local and non-local
- The core mechanisms are **the same** cross-domain
- That is: linguistic dependencies are **local** over the right **structural representations**

Some Insights

Parallels between phonology and syntax?

- What would a computational linguist tell you?
  Well, it depends!
- What will I show you?
  They are fundamentally similar!

The Take-Home Message

- **Two kind of dependencies**: local and non-local
- The core mechanisms are **the same** cross-domain
- That is: linguistic dependencies are **local** over the right **structural representations**

Parallels between Phonology and Syntax

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

Parallels between Phonology and Syntax

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 patters 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 patters 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: *[+voice]$
- **German**: ***z**$, ***v**$,***d**$ ($ = word edge).

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

Example: Intervocalic voicing

- Forbid voicess segments in-between two vowels: *V[-voice]V
- **German**: *a**s**e, *i**s**e, *e**s**e, *i**s**i, ...

$ f a **s** e r $ $ f a z e r $

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 $\boxed{\textbf{d}\ \$}$ $\quad$ ^{ok} $ r a $\boxed{t\ \$}$

Example: Intervocalic voicing

- Forbid voicess segments in-between two vowels: *V[-voice]V
- **German**: *a**s**e, *i**s**e, *e**s**e, *i**s**i, . . .

$ f a **s** e r $ $\qquad$ $ f a z e r $

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 $\boxed{\textbf{d} \ \$}$ $\quad$ ^{ok} $ r a $\boxed{\text{t} \ \$}$

Example: Intervocalic voicing

- Forbid voicess segments in-between two vowels: *V[-voice]V
- **German**: *a**s**e, *i**s**e, *e**s**e, *i**s**i, . . .

* $ f $\boxed{\text{a} \ \textbf{s} \ \text{e}}$ r $ $\quad$ ^{ok} $ f $\boxed{\text{a} \ \text{z} \ \text{e}}$ 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
$\mathrm{N}^+\ \mathrm{D}^+\ \mathrm{D}^-$	N^-

Local Dependencies in Syntax

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

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

Merge

's $\mathrm{N}^+\ \mathrm{D}^+\ \mathrm{D}^-$

cat N^-

Local Dependencies in Syntax

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

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

Mary
D^-

Merge

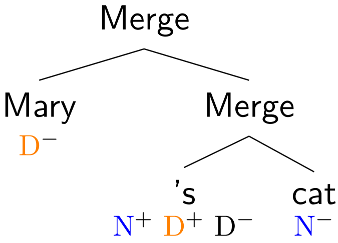
's
N^+ D^+ D^-

cat
N^-

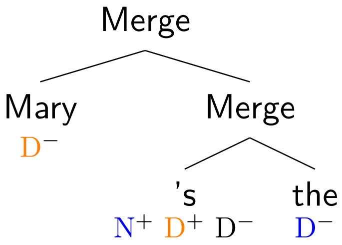
Local Dependencies in Syntax

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

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

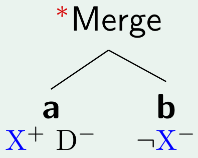


Merge is SL (Graf 2012)

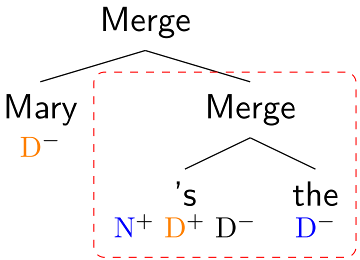


SL constraints on Merge

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

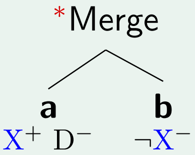


Merge is SL (Graf 2012)



SL constraints on Merge

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



Interim Summary

	Local	Data Structure
Phonology	?	?
Syntax	?	?

Local phenomena modeled by n-grams of bounded size:

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

Interim Summary

	Local	Data Structure
Phonology	**SL**	**Strings**
Syntax	**SL**	**Trees**

Local phenomena modeled by n-grams of bounded size:

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

Interim Summary

	Local	Non-local	Data Structure
Phonology	**SL**	?	**Strings**
Syntax	**SL**	?	**Trees**

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) a. * ha**s**xintilawaʃ
  b. * haʃxintilawa**s**
  c. haʃxintilawaʃ

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

  (4) a. **LHLLLL**
  b. **LLLLHL**
  c. * **LHLLHL**
  d. **LHHHHL**

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. * ha**s**xintilawaʃ
b. * haʃxintilawa**s**
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. * ha**s**xintilawaʃ

b. * haʃxintilawa**s**

c. haʃxintilawaʃ

Example: Samala

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

\$ 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. * ha**s**xintilawaʃ
b. * haʃxintilawa**s**
c. haʃxintilawaʃ

Example: Samala

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

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

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

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

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

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

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

* \$ s t a j a n o w o n w a ʃ \$

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

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

* $ h a s x i n t i l a w a ʃ $

$ h a ʃ x i n t i l a w a ʃ $

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

* $ s t a j a n o w o n w a ʃ $

Locality Over Tiers

$*\$ \boxed{\textbf{s}\ t\ a\ j\ a\ n\ o\ w\ o\ n\ w\ a\ \int} \$$

- 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

$$* \$ \boxed{s\ t\ a\ j\ a\ n\ o\ w\ o\ n\ w\ a\ ʃ} \$$$

- 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

* $ h a s x i n t i l a w ʃ $ ^{ok} $ h a ʃ x i n t i l a w 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]

Example: TSL Samala

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

Unbounded Dependencies are TSL

- Let's revisit Samala Sibilant Harmony

(6) a. * ha**s**xintilawaʃ

b. * haʃxintilawa**s**

c. haʃxintilawaʃ

- 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 ʃ $ ^{ok} $ h a ʃ x i n t i l a w 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]

Example: TSL Samala

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

Example: TSL Samala

s

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

ok $ h a ʃ x i n t i l a w 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]

Example: TSL Samala

s

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

Example: TSL Samala

s

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

ok $ h a ʃ x i n t i l a w ʃ $

Unbounded Dependencies are TSL

- Let's revisit Samala Sibilant Harmony

(6) a. * ha**s**xintilawaʃ
b. * haʃxintilawa**s**
c. haʃxintilawaʃ

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

Example: TSL Samala

s

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

Unbounded Dependencies are TSL

- Let's revisit Samala Sibilant Harmony

(6) a. * ha**s**xintilawaʃ

b. * haʃxintilawa**s**

c. haʃxintilawaʃ

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

Example: TSL Samala

s

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

Example: TSL Samala

s

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

ok $ h a ʃ x i n t i l a w ʃ $

Unbounded Dependencies are TSL

- Let's revisit Samala Sibilant Harmony

(6) a. * ha**s**xintilawaʃ

b. * haʃxintilawa**s**

c. haʃxintilawaʃ

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

Example: TSL Samala

s

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

Example: TSL Samala

s

* $ h a s x i n t i l a w ʃ $ ^{ok} $ h a ʃ x i n t i l a w 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]

Example: TSL Samala

s

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

^{ok} $ h a ʃ x i n t i l a w 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]

Example: TSL Samala

s

* $ h a s x i n t i l a w ʃ $ ^{ok} $ h a ʃ x i n t i l a w 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]

Example: TSL Samala

s ʃ

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

Unbounded Dependencies are TSL

- Let's revisit Samala Sibilant Harmony

(6) a. * ha**s**xintilawaʃ

b. * haʃxintilawa**s**

c. haʃxintilawaʃ

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

I.E. *sʃ, *sʒ, *zʃ, *zʒ, *ʃs, *ʒs, *ʃz, *ʒz

Example: TSL Samala

s ʃ

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

Unbounded Dependencies are TSL

- Let's revisit Samala Sibilant Harmony

(6) a. * ha**s**xintilawaʃ

b. * haʃxintilawa**s**

c. haʃxintilawaʃ

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

  I.E. *sʃ, *sʒ, *zʃ, *zʒ, *ʃs, *ʒs, *ʃz, *ʒz

Example: TSL Samala

s ʃ

* $ h a **s** x i n t i l a w ʃ $

ok $ h a ʃ x i n t i l a w ʃ $

Unbounded Dependencies are TSL

- Let's revisit Samala Sibilant Harmony

(6) a. * ha**s**xintilawaʃ

b. * haʃxintilawa**s**

c. haʃxintilawaʃ

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

I.E. *sʃ, *sʒ, *zʃ, *zʒ, *ʃs, *ʒs, *ʃz, *ʒz

Example: TSL Samala

s ʃ ʃ ʃ

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

Unbounded Dependencies are TSL

- Let's revisit Samala Sibilant Harmony

(6)
a. * ha**s**xintilawaʃ
b. * haʃxintilawa**s**
c. haʃxintilawaʃ

- What do we need to project? [+strident]
- What do we need to ban? *[+ant][−ant],*[−ant][+ant]
  I.E. *sʃ, *sʒ, *zʃ, *zʒ, *ʃs, *ʒs, *ʃz, *ʒz

Example: TSL Samala

$\boxed{\text{s} \quad \text{ʃ}}$

* $ h a **s** x i n t i l a w ʃ $

$\boxed{\text{ʃ} \quad \text{ʃ}}$

ok $ h a ʃ x i n t i l a w ʃ $

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. **LHLLLL**
b. **LLLLHL**
c. * **LHLLHL**
d. **LHHHHL**

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. **LHLLLL**
  b. **LLLLHL**
  c. * **LHLLHL**
  d. **LHHHHL**

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. **LHLLLL**
b. **LLLLHL**
c. * **LHLLHL**
d. **LHHHHL**

Example

*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. **LHLLLL**
b. **LLLLHL**
c. ***LHLLHL**
d. **LHHHHL**

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. **LHLLLL**
b. **LLLLHL**
c. * **LHLLHL**
d. **LHHHHL**

Example

$\boxed{\text{L H L L H}}$ 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

^{ok}**L H L L L 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

H

....

ok **L H** L L L 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

H L

^{ok}L H L L L 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

H L

^{ok}**L H $\boxed{\textbf{L L}}$ L L** $\quad\quad$ ***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

H L

^{ok}**L H L $\boxed{\textbf{L L}}$ 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

H L

^{ok}**L H L L $\boxed{\text{L L}}$** ***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

$\boxed{\textbf{H L}}$

^{ok}**L H L L L 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

$\boxed{\textbf{H L}}$

^{ok}**L H L L L 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

H L	**H**
^{ok}**L H L L L 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

H L	**H L**
^{ok}**L H L L L 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

$\boxed{\text{H L}}$	H L
....	....
^{ok}L H L L L L	*L H $\boxed{\text{L L}}$ 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	**H L H**
^{ok}**L H L L L 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

HL	HL HL
^{ok}LHLLLL	*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**

Example

H L	**H L H L**
^{ok}**L H L L L 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

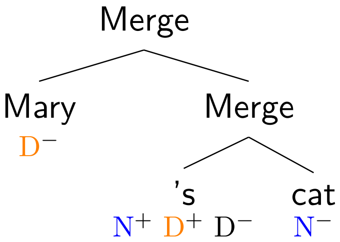
H L	**H L H L**
^{ok}**L H L L L L**	***L H L L H L**

- Most non-local dependencies in phonology are TSL
- **What about syntax?**

Non-Local Dependencies in Syntax

Let's stick to core operations:

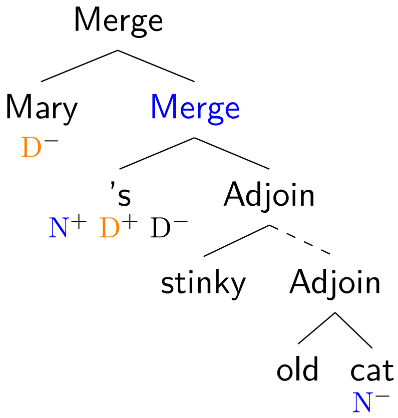
- Move
- Merge?



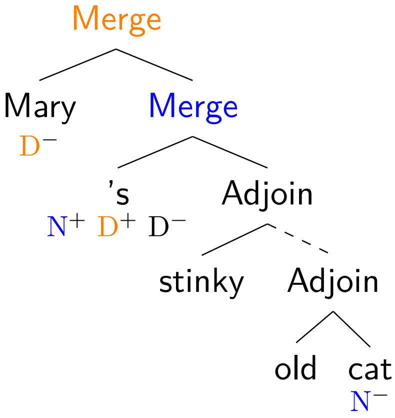
Non-Local Dependencies in Syntax

Let's stick to core operations:

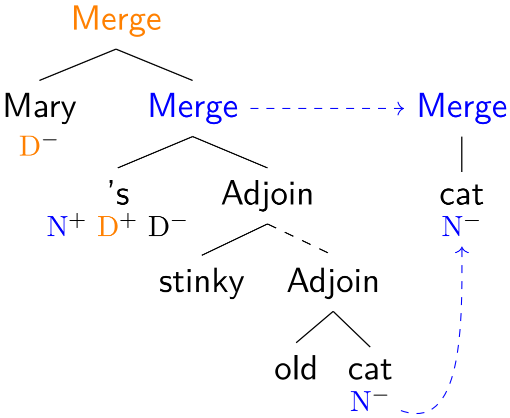
- Move
- **Merge**: Unbounded adjunction
  Frey and Gärtner (2002); Graf (2017)



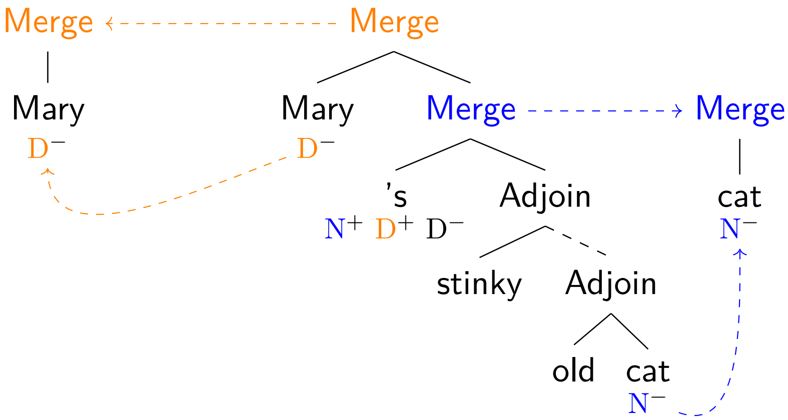
TSL over Trees: Projecting Tiers



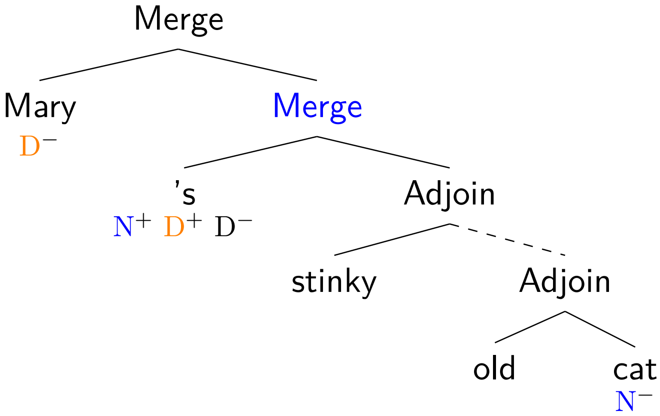
TSL over Trees: Projecting Tiers



TSL over Trees: Projecting Tiers

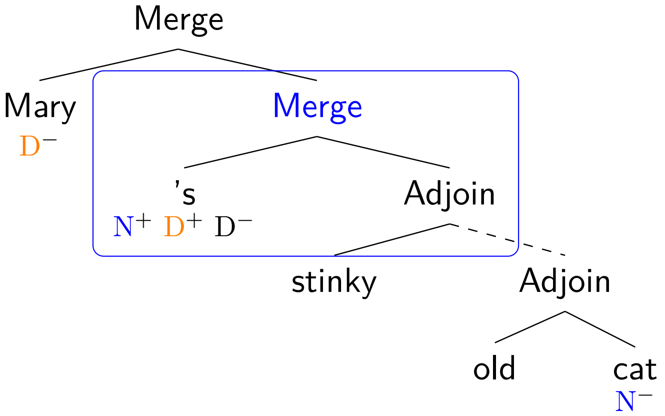


Merge with Adjunction is TSL



A TSL grammar for Merge

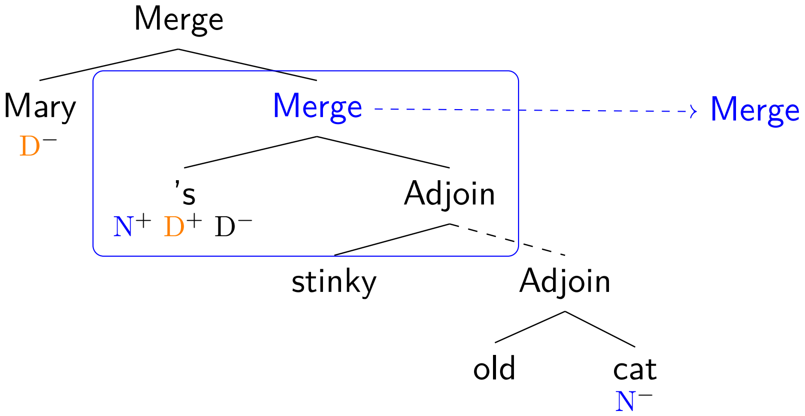
Merge with Adjunction is TSL



A TSL grammar for Merge

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

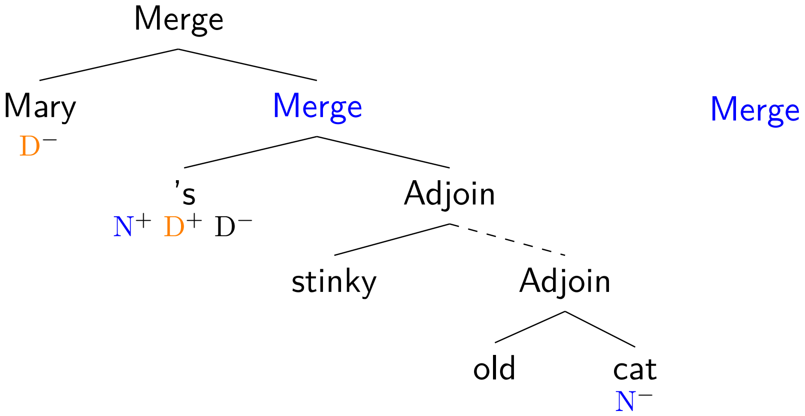
Merge with Adjunction is TSL



A TSL grammar for Merge

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

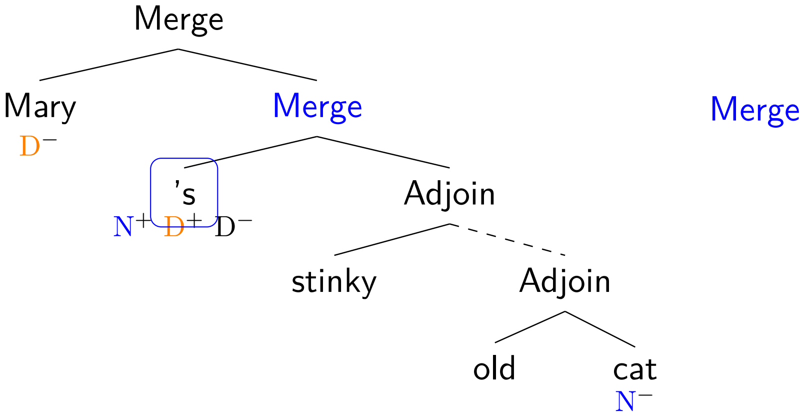
Merge with Adjunction is TSL



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. $X = N$)

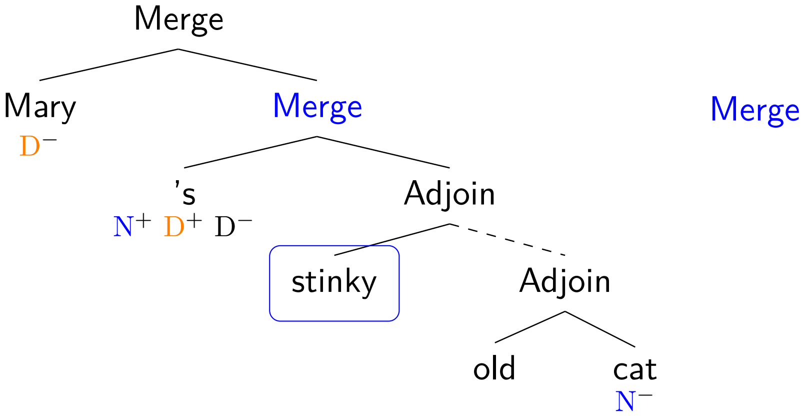
Merge with Adjunction is TSL



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. $X = N$)

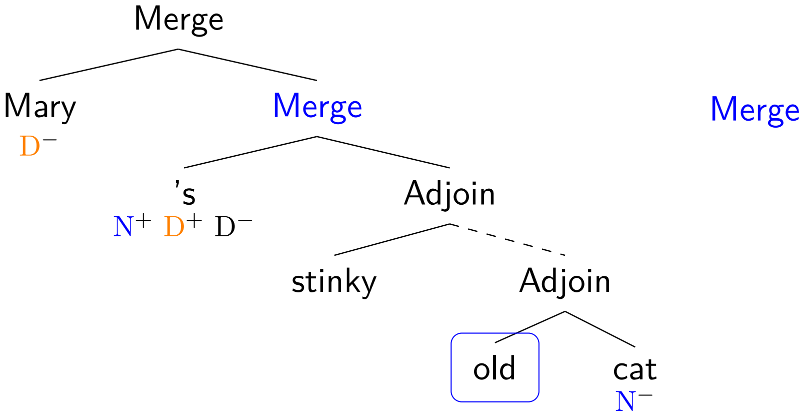
Merge with Adjunction is TSL



A TSL grammar for Merge

1. Project **Merge** iff a child has X^+ (e.g. $\mathrm{X} = \mathrm{N}$)
2. Project any node which has X^- (e.g. $\mathrm{X} = \mathrm{N}$)

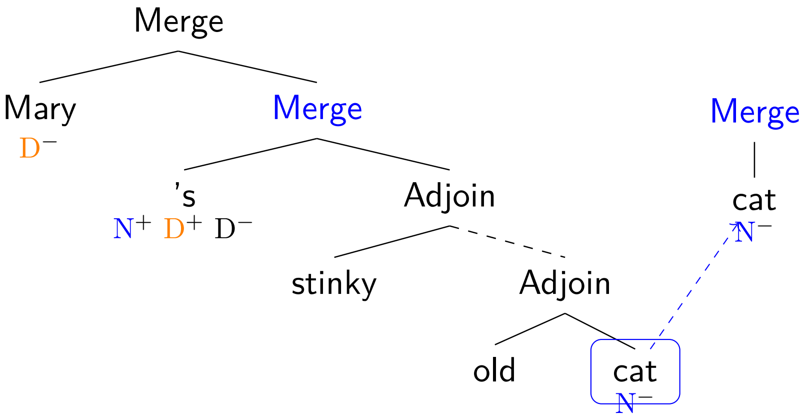
Merge with Adjunction is TSL



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. $X = N$)

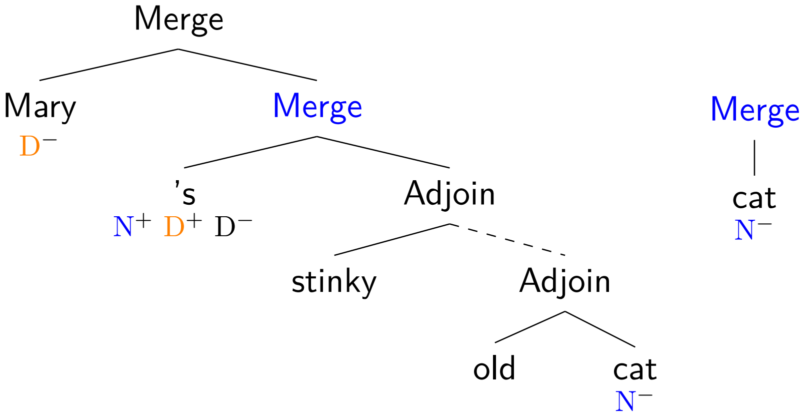
Merge with Adjunction is TSL



A TSL grammar for Merge

1. Project **Merge** iff a child has X^+ (e.g. $\mathrm{X} = \mathrm{N}$)
2. Project any node which has X^- (e.g. $\mathrm{X} = \mathrm{N}$)

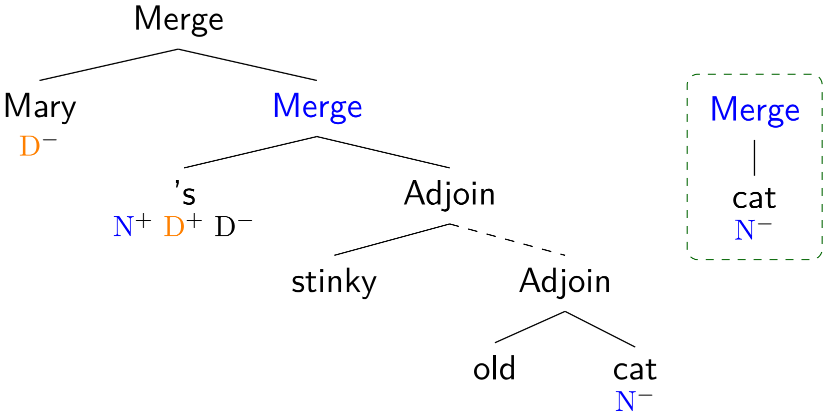
Merge with Adjunction is TSL



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. $X = N$)

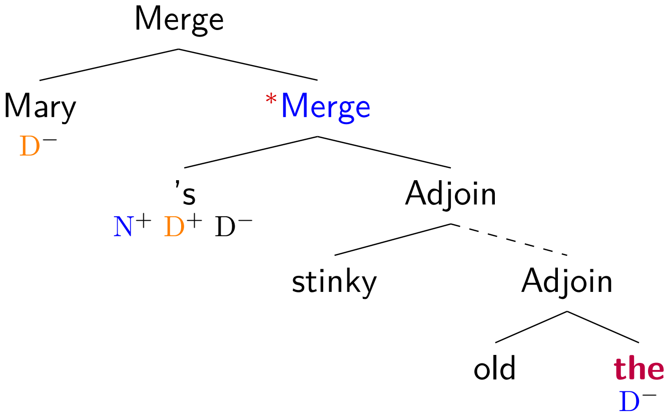
Merge with Adjunction is TSL



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. $X = N$)
3. No Merge without exactly one LI among its daughters.

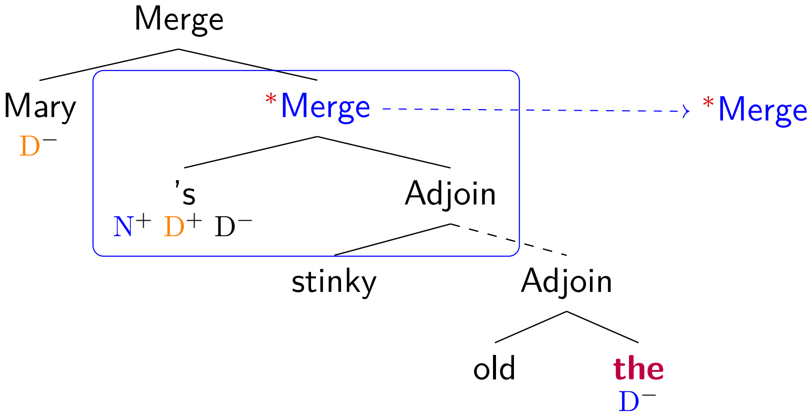
Merge with Adjunction is TSL



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.

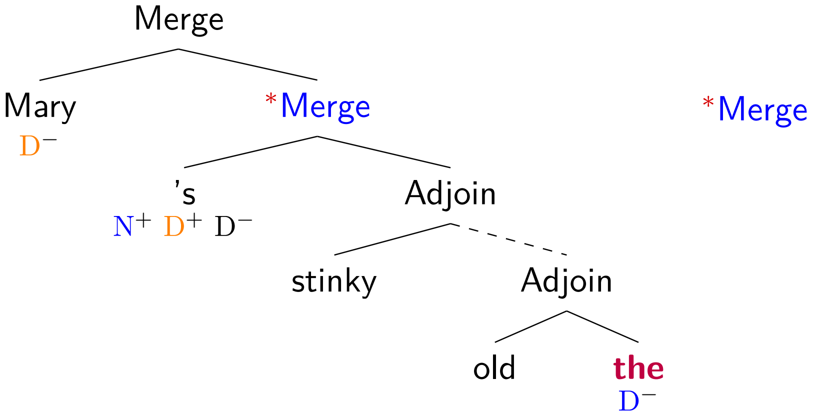
Merge with Adjunction is TSL



A TSL grammar for Merge

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

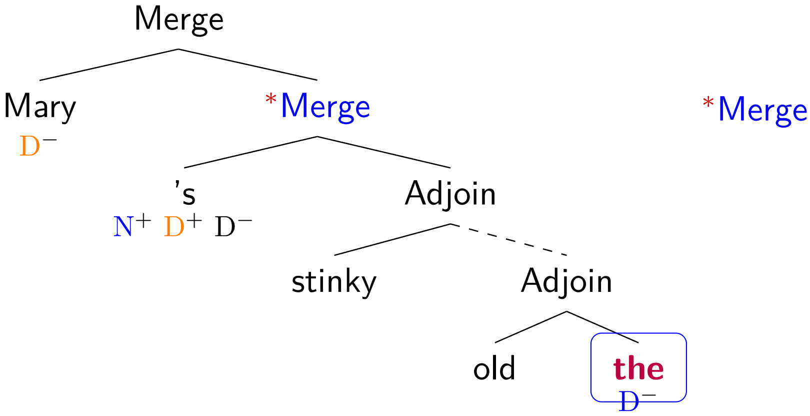
Merge with Adjunction is TSL



A TSL grammar for Merge

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

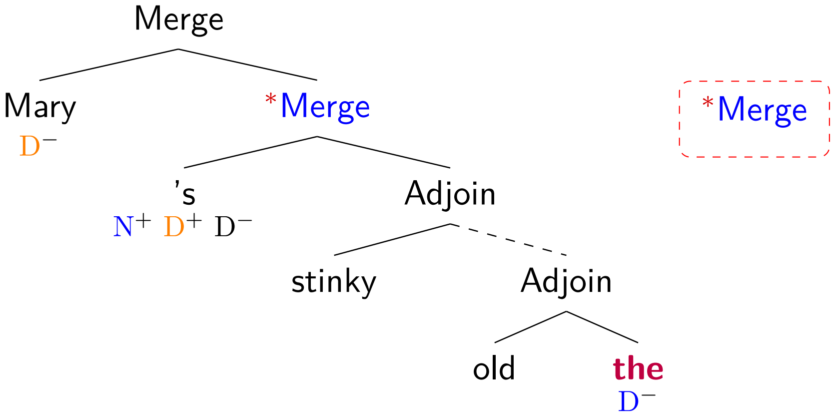
Merge with Adjunction is TSL



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. $\mathrm{X} = \mathrm{V}$)
2. Project any node which has X^+ (e.g. $\mathrm{X} = \mathrm{V}$)
3. No Merge without exactly one LI among its daughters.

Parallels Between Phonology And Syntax

	Local	**Non-local**
Phonology	?	?
Syntax	?	?

- **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

	Local	Non-local
Phonology	**SL**	?
Syntax	**SL**	?

- **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

	Local	Non-local
Phonology	**SL**	**TSL**
Syntax	**SL**	**TSL**

- **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

	Local	Non-local	Data Structure
Phonology	SL	TSL	Strings
Syntax	SL	TSL	Trees

- **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

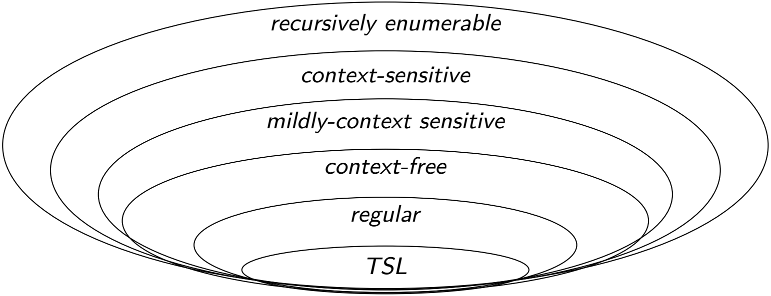
	Local	Non-local	Data Structure
Phonology	SL	TSL	Strings
Syntax	SL	TSL	Trees

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

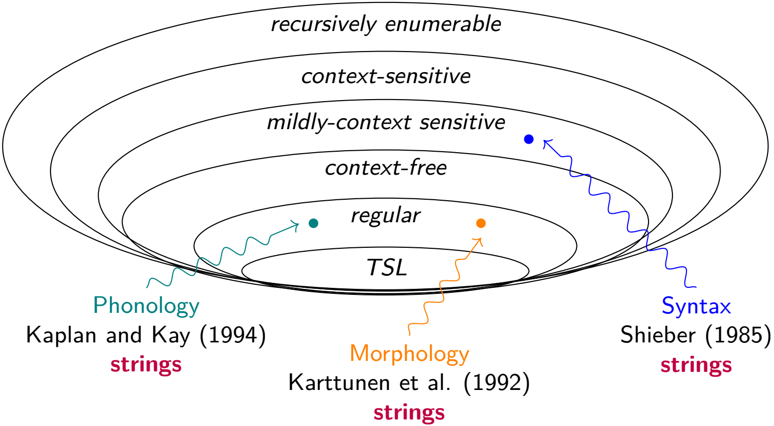
Strong Cognitive Parallelism Hypothesis

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

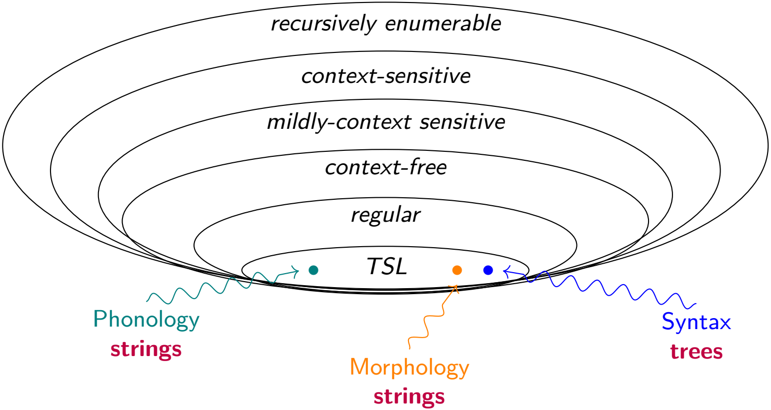
A Bird's-Eye View of the Framework



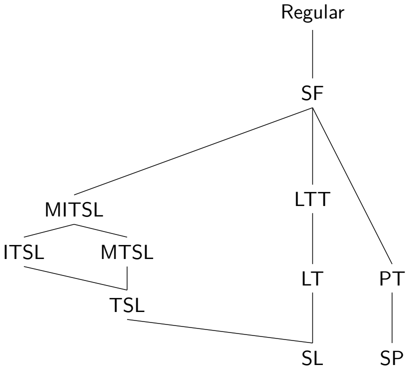
A Bird's-Eye View of the Framework



A Bird's-Eye View of the Framework



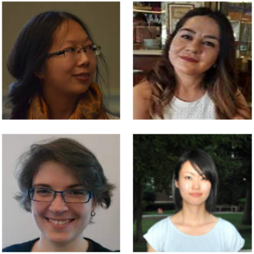
Refining the Hierarchy via Typological Insights



- The goal is not identifying a single "correct" class
- Pinpoint fundamental properties of the patterns: SL: $\triangleleft$, TSL: $\triangleleft_T$, etc

Syntax beyond Merge and Move

- regular tree languages
  (Michaelis 2004; Kobele et al. 2007)
- subregular operations (Graf 2018)
- subregular dependencies/constraints
  (Vu et al. 2019; Shafiei and Graf 2019)
- tree automata and parsing restrictions
  (Graf & De Santo 2020)



Interim Summary: Again, So What?

Strong Parallelism Hypothesis

Dependencies in phonology, (morphology), and syntax are **subregular** over their respective **structural representations**.

We gain a unified perspective on:

- Attested and unattested typology
- learnability?
- cognition

Interim Summary: Again, So What?

Strong Parallelism Hypothesis

Dependencies in phonology, (morphology), and syntax are **subregular** over their respective **structural representations**.

We gain a unified perspective on:

- Attested and unattested typology
  - × Intervocalic Voicing iff applied an even times in the string
  - × Have a CP iff it dominates ≥ 3 TPs
- learnability?
- cognition

Interim Summary: Again, So What?

Strong Parallelism Hypothesis

Dependencies in phonology, (morphology), and syntax are **subregular** over their respective **structural representations**.

We gain a unified perspective on:

- Attested and unattested 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

Interim Summary: Again, So What?

Strong Parallelism Hypothesis

Dependencies in phonology, (morphology), and syntax are **subregular** over their respective **structural representations**.

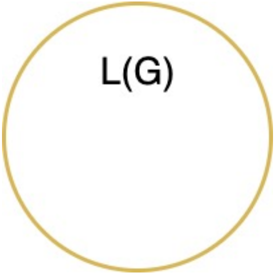
We gain a unified perspective on:

- Attested and unattested 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 ?

Outline

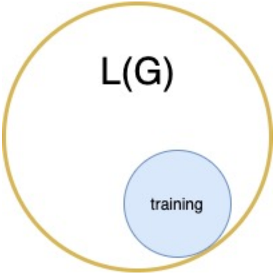
Artificial Grammar Learning (AGL)

- Can be used to test implicit learning abilities (Reber, 1976)



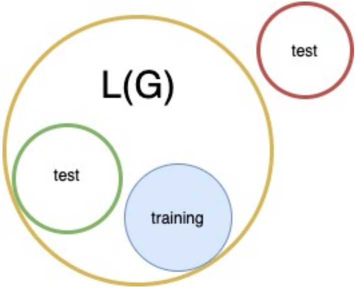
Artificial Grammar Learning (AGL)

- Can be used to test implicit learning abilities (Reber, 1976)



Artificial Grammar Learning (AGL)

- Can be used to test implicit learning abilities (Reber, 1976)



Reber (1976)

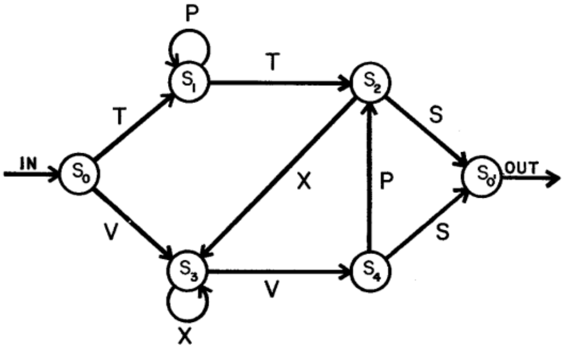


Fig. 1. Schematic state diagram of the grammar used to generate the grammatical stimulus items.

- Stimuli generated from an FST or randomly
  - 28 sentences per group, in sets of four sentences each
  - Participants asked to reproduce the sentences in a group
  - Participants informed of correct/incorrect reproductions, but not of error type

Reber (1976) [cont.]

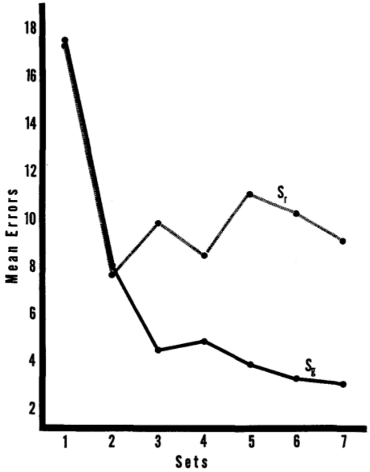


Fig. 2. Mean number of errors to criterion on each of the seven learning sets.

- Stimuli generated from an FST or randomly
  - Significant differences between learning trajectories across participant group

Testing Subregular Predictions

complexity

		Regular	Monadic Second-Order Logic
		$\cup$	
Locally Threshold Testable	$\subset$	Star Free	First-Order Logic
$\cup$		$\cup$	
Locally Testable		Piecewise Testable	Propositional Logic
$\cup$		$\cup$	
Strictly Local	$\subset$ TSL	Strictly Piecewise	Conjunction of Negative Literals
$S/\lhd$		$< / \lhd^+$	

Example: Attested vs. Unattested Patterns

Attested: Unbounded Sibilant Harmony

- Every sibilant needs to harmonize

s ʃ   ʃ ʃ

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

Unattested: First-Last Harmony

- Harmony only holds between initial and final segments

s ʃ   s ʃ

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

Lai (2015)

Learnable vs. Unlearnable Harmony Patterns

Regine Lai

Posted Online July 09, 2015
https://doi.org/10.1162/LING_a_00188

Linguistic Inquiry
Volume 46 | Issue 3 | Summer 2015
p.425-451

Keywords: phonotactics, learnability, computational phonology, formal theory, typology, dependencies

Lai (2015): Stimuli

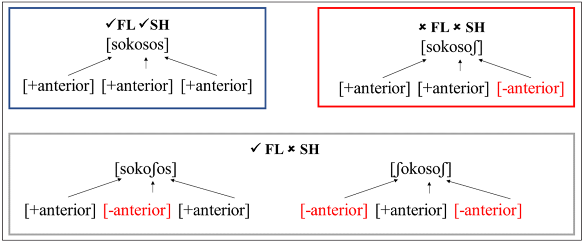


Figure 3: Comparison of SH and FL stimuli.

Lai (2015): Stimuli

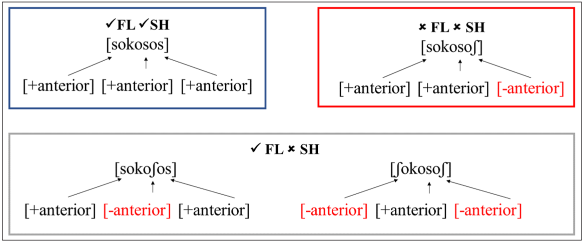
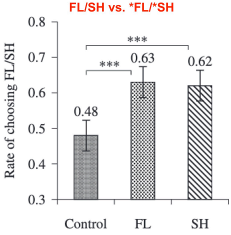


Figure 3: Comparison of SH and FL stimuli.

Table 6
Predicted results with respect to the control group for each test pairing if Sibilant Harmony and First-Last Assimilation grammars were internalized

Conditions	Pairs: FL/*SH vs. *FL/*SH (e.g., [s . . . ʃ . . . s] vs. [s . . . s . . . ʃ]) Rate of FL/*SH	FL/SH vs. *FL/*SH (e.g., [s . . . s . . . s] vs. [s . . . s . . . ʃ]) Rate of FL/SH	FL/SH vs. FL/*SH (e.g., [s . . . s . . . s] vs. [s . . . ʃ . . . s]) Rate of FL/SH
SH	~ Control	> Control	> Control
FL	> Control	> Control	~ Control

Lai (2015): Results



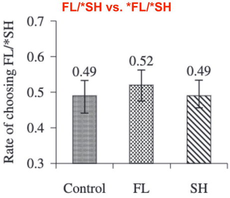


Table 6
Predicted results with respect to the control group for each test pairing if Sibilant Harmony and First-Last Assimilation grammars were internalized

	Pairs		
Conditions	FL/*SH vs. *FL/*SH (e.g., [s . . . ʃ . . . s] vs. [s . . . s . . . ʃ]) Rate of FL/*SH	FL/SH vs. *FL/*SH (e.g., [s . . . s . . . s] vs. [s . . . s . . . ʃ]) Rate of FL/SH	FL/SH vs. FL/*SH (e.g., [s . . . s . . . s] vs. [s . . . ʃ . . . s]) Rate of FL/SH
SH	∼ Control	> Control	> Control
FL	> Control	> Control	∼ Control

- See Avcu and Hestvik (2020), Avcu et al. (2019) for replications

Lai (2015): Results

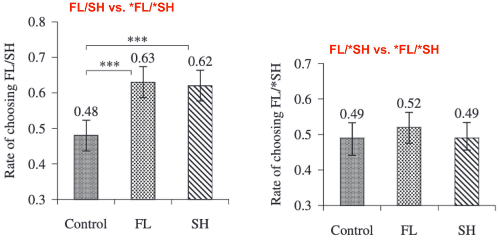
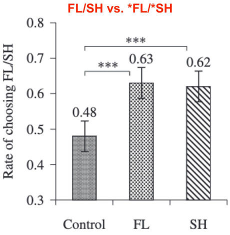


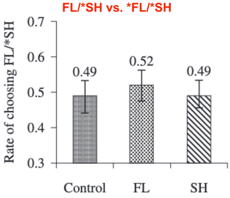
Table 6
Predicted results with respect to the control group for each test pairing if Sibilant Harmony and First-Last Assimilation grammars were internalized

	Pairs		
Conditions	FL/*SH vs. *FL/*SH (e.g., [s . . . ʃ . . . s] vs. [s . . . s . . . ʃ]) Rate of FL/*SH	FL/SH vs. *FL/*SH (e.g., [s . . . s . . . s] vs. [s . . . s . . . ʃ]) Rate of FL/SH	FL/SH vs. FL/*SH (e.g., [s . . . s . . . s] vs. [s . . . ʃ . . . s]) Rate of FL/SH
SH	∼ Control	> Control	> Control
FL	> Control	> Control	∼ Control

- See Avcu and Hestvik (2020), Avcu et al. (2019) for replications

Lai (2015): Full Results





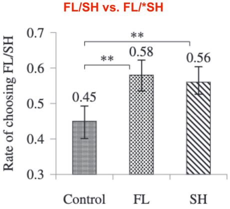
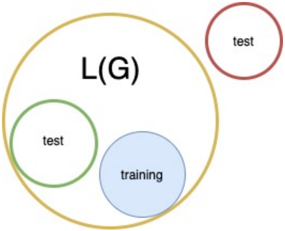


Table 6
Predicted results with respect to the control group for each test pairing if Sibilant Harmony and First-Last Assimilation grammars were internalized

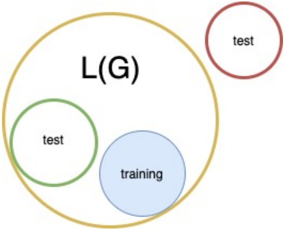
	Pairs		
Conditions	FL/*SH vs. *FL/*SH (e.g., [s . . . ʃ . . . s] vs. [s . . . s . . . ʃ]) Rate of FL/*SH	FL/SH vs. *FL/*SH (e.g., [s . . . s . . . s] vs. [s . . . s . . . ʃ]) Rate of FL/SH	FL/SH vs. FL/*SH (e.g., [s . . . s . . . s] vs. [s . . . ʃ . . . s]) Rate of FL/SH
SH	∼ Control	> Control	> Control
FL	> Control	> Control	∼ Control

Testing Predictions with AGL



- It is a powerful technique
- Careful in drawing inferences from laboratory behavior
- Importantly: Common fallacies in experimental design

Testing Predictions with AGL



- It is a powerful technique
- Careful in drawing inferences from laboratory behavior
- Importantly: Common fallacies in experimental design

Generalizability in AGL

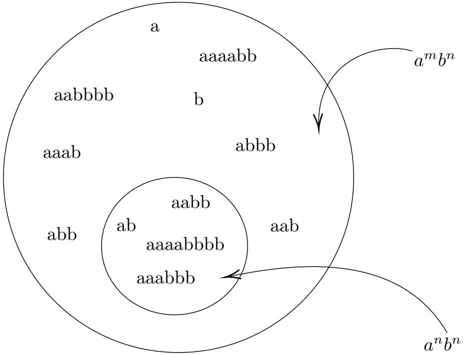
A famous CFL exemplar: $A^n B^n$

$$ab, aabb, aaabbb, aaaabbbb, \ldots$$

Generalizability in AGL

A famous CFL exemplar: A^nB^n

$$ab, aabb, aaabbb, aaaabbbb, \ldots$$



Evaluating Contrasts (1/5)

A famous CFL exemplar: $A^n B^n$

$$ab, aabb, aaabbb, aaaabbbb, \ldots$$

Which features might one generalize to?

- All As precede all Bs
- Strings are all of even length
- $|w|_A = |w|_B$
- ...

Picking the right contrasts is essential!

Evaluating Contrasts (1/5)

A famous CFL exemplar: A^nB^n

$$ab, aabb, aaabbb, aaaabbbb, \dots$$

Which features might one generalize to?

- All As precede all Bs **(SL)**
- Strings are all of even length **(REG)**
- $|w|_A = |w|_B$ **(CF)**
- ...

Picking the right contrasts is essential!

Evaluating Contrasts (1/5)

A famous CFL exemplar: A^nB^n

$$ab, aabb, aaabbb, aaaabbbb, \ldots$$

Which features might one generalize to?

- All As precede all Bs **(SL)**
- Strings are all of even length **(REG)**
- $|w|_A = |w|_B$ **(CF)**
- ...

Picking the right contrasts is essential!

Evaluating Contrasts (2/5)

A famous CFL exemplar: $A^n B^n$

$$ab, aabb, aaabbb, aaaabbbb, \ldots$$

Which features might one generalize to?

- **All As precede all Bs** **(SL)**
- Strings are all of even length **(REG)**
- $|w|_A = |w|_B$ **(CF)**

AAABBB ABABAB

Evaluating Contrasts (3/5)

A famous CFL exemplar: $A^n B^n$

$$ab, aabb, aaabbb, aaaabbbb, \ldots$$

Which features might one generalize to?

- All As precede all Bs **(SL)**
- **Strings are all of even length** **(REG)**
- $|w|_A = |w|_B$ **(CF)**

AAABBB AABBB

Evaluating Contrasts (4/5)

A famous CFL exemplar: $A^n B^n$

$$ab, aabb, aaabbb, aaaabbbb, \ldots$$

Which features might one generalize to?

- All As precede all Bs **(SL)**
- Strings are all of even length **(REG)**
- $|w|_A = |w|_B$ **(CF)**

AAABBB AABBBB

Evaluating Contrasts (5/5)

A famous CFL exemplar: $A^n B^n$

$$ab, aabb, aaabbb, aaaabbbb, \ldots$$

Which features might one generalize to?

- All As precede all Bs: ABA **(SL)**
- Strings are all of even length: AABBB **(REG)**
- $|w|_A = |w|_B$: ABAB **(CF)**

Evaluating Contrasts (5/5)

A famous CFL exemplar: $A^n B^n$

$$ab, aabb, aaabbb, aaaabbbb, \ldots$$

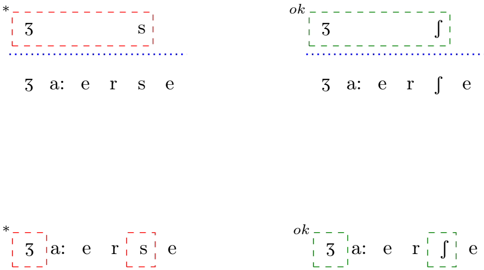
Which features might one generalize to?

- All As precede all Bs: ABA **(SL)**
- Strings are all of even length: AABBB **(REG)**
- $|w|_A = |w|_B$: ABAB **(CF)**
- finite bound
- ...

AAABBB AAAABBBB

Evaluating Contrasts: Picking the Right Primitives

Long-distance relations?



- Stimuli are often ambiguous between overlapping classes
- Distinguishing between representation requires care

Evaluating Contrasts: Picking the Right Primitives

Long-distance relations?

* ʒ s

ʒ aː e r s e

ok ʒ ʃ

ʒ aː e r ʃ e

* ʒ aː e r s e

ok ʒ aː e r ʃ e

- Stimuli are often ambiguous between overlapping classes
- Distinguishing between representation requires care

AGL and Syntax/Semantics

> *distinctions between mechanisms for recognizing non-Finite-State stringsets depend on the way in which the additional structure, beyond the string itself, is organized; these are issues that show up in the analysis of the string, not in its form as a sequence of events.*
>
> *Rogers & Pullum 2011*

In other words:

- Questions of complexity confounded by representations
- Questions of representations confounded by procedures

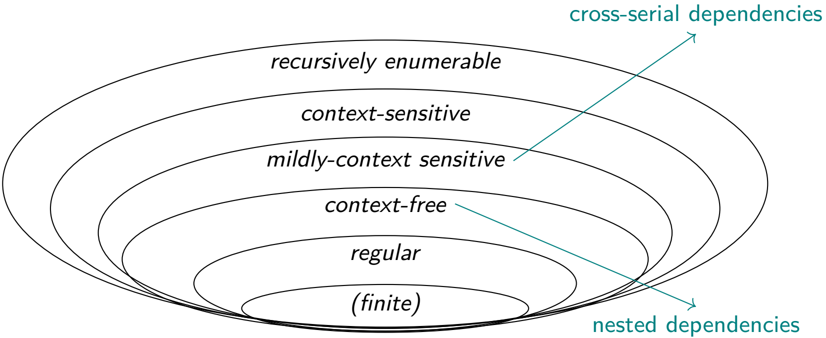
AGL and Syntax/Semantics

> *distinctions between mechanisms for recognizing non-Finite-State stringsets depend on the way in which the additional structure, beyond the string itself, is organized; these are issues that show up in the analysis of the string, not in its form as a sequence of events.*
>
> *Rogers & Pullum 2011*

In other words:

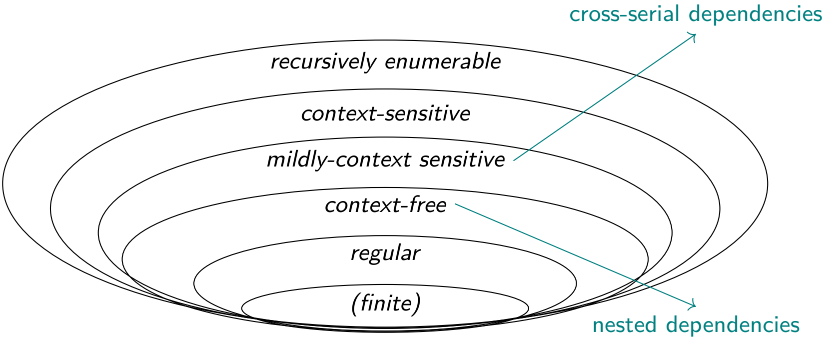
- Questions of complexity confounded by representations
- Questions of representations confounded by procedures

Syntactic Expressivity



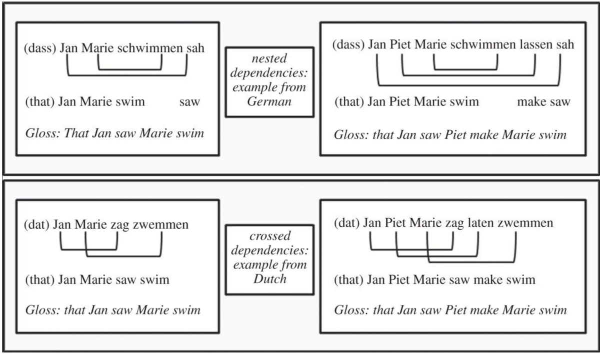
- cross-serial preferred over nested (Bach et al. 1986)
- against predictions from the CH?
  (Chesi & Moro 2014; de Vries et al. 2012)

Syntactic Expressivity



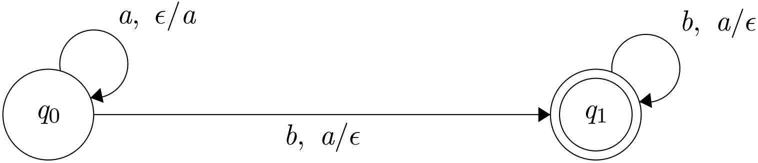
- cross-serial preferred over nested (Bach et al. 1986)
- against predictions from the CH?
  (Chesi & Moro 2014; de Vries et al. 2012)

Expressivity vs. Procedures



- cross-serial preferred over nested (Bach et al. 1986)
- against predictions from the CH?
  (Chesi & Moro 2014; de Vries et al. 2012)
- BUT: this can easily be derived via processing mechanisms
  (Savitch 1989; Joshi, 1990; Rainbow and Joshi,1994)
- recognition complexity requires a precise theory of parsing cost

AGL and Syntax/Semantics [cont.]

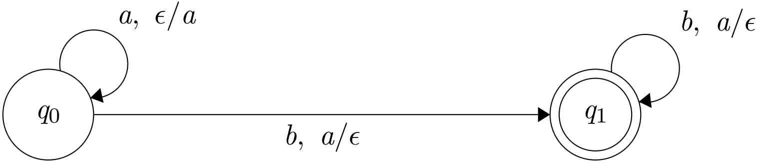


- A^nB^n does not necessarily imply a proper stack
  a PDA with a single counter is enough (Counter Machines)
- Same for the language of strings of **well-nested parentheses**
- Phrase-structure analyses often depend on distinctions based on the meaning of the strings

Complicated questions:

- What **representations** are relevant?
- How are they connected to **tasks**?
- How do we probe them?

AGL and Syntax/Semantics [cont.]

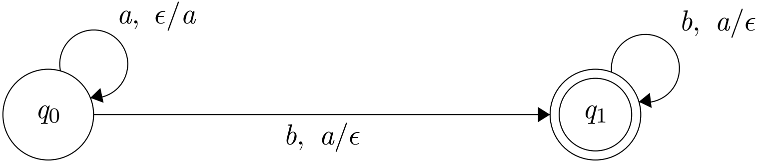


- A^nB^n does not necessarily imply a proper stack
  a PDA with a single counter is enough (Counter Machines)
- Same for the language of strings of **well-nested parentheses**
- Phrase-structure analyses often depend on distinctions based on the meaning of the strings

Complicated questions:

- What **representations** are relevant?
- How are they connected to **tasks**?
- How do we probe them?

AGL and Syntax/Semantics [cont.]

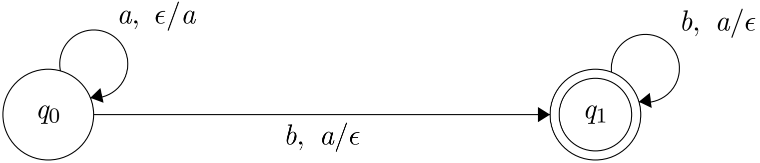


- A^nB^n does not necessarily imply a proper stack
  a PDA with a single counter is enough (Counter Machines)
- Same for the language of strings of **well-nested parentheses**
- Phrase-structure analyses often depend on distinctions based on the meaning of the strings

Complicated questions:

- What **representations** are relevant?
- How are they connected to **tasks**?
- How do we probe them?

AGL and Syntax/Semantics [cont.]



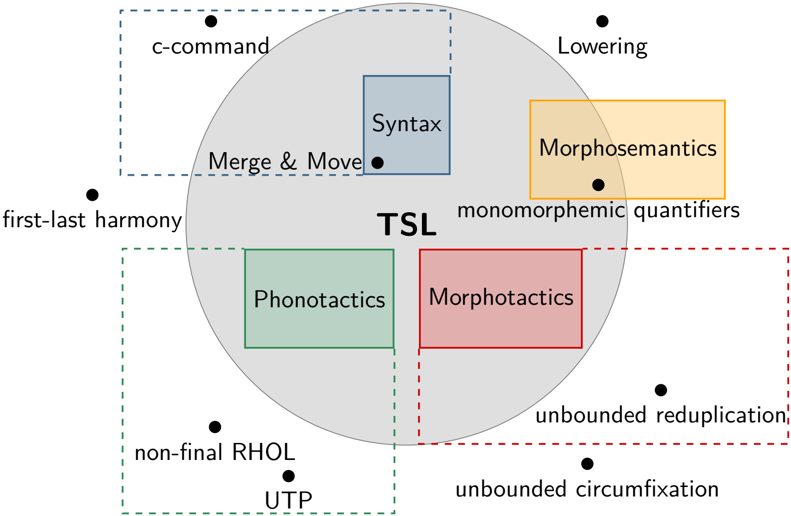
- A^nB^n does not necessarily imply a proper stack
  a PDA with a single counter is enough (Counter Machines)
- Same for the language of strings of **well-nested parentheses**
- Phrase-structure analyses often depend on distinctions based on the meaning of the strings

Complicated questions:

- What **representations** are relevant?
- How are they connected to **tasks**?
- How do we probe them?

Outline

Subregularity Across Modules



In a Nutshell

Generalized Quantifiers and Semantic Complexity

Semantic automata (SA) as a model of quantifiers' verification

- insights into quantifiers' interpretation
- link between formal language theory and model theory

Beyond the SA perspective

- Formal language theory is richer that automata theory
- Coming back to formal language theory
  $\rightarrow$ subregular hierarchy & quantifier languages
  (De Santo et al. 2017; Graf 2019)

Consequences

- complexity independent of the recognition mechanism
- cross-domain parallels, cognitive predictions, …

In a Nutshell

Generalized Quantifiers and Semantic Complexity

Semantic automata (SA) as a model of quantifiers' verification

- insights into quantifiers' interpretation
- link between formal language theory and model theory

Beyond the SA perspective

- Formal language theory is richer that automata theory
- Coming back to formal language theory
  $\rightarrow$ subregular hierarchy & quantifier languages
  (De Santo et al. 2017; Graf 2019)

Consequences

- complexity independent of the recognition mechanism
- cross-domain parallels, cognitive predictions, ...

In a Nutshell

Generalized Quantifiers and Semantic Complexity

Semantic automata (SA) as a model of quantifiers' verification

- insights into quantifiers' interpretation
- link between formal language theory and model theory

Beyond the SA perspective

- Formal language theory is richer that automata theory
- Coming back to formal language theory
  $\rightarrow$ subregular hierarchy & quantifier languages
  (De Santo et al. 2017; Graf 2019)

Consequences

- complexity independent of the recognition mechanism
- cross-domain parallels, cognitive predictions, ...

Generalized Quantifiers

Generalized quantifier $\mathbf{Q}(\mathbf{A}, \mathbf{B})$:

- two sets **A** and **B** as arguments
- returns truth value $(0, 1)$

Example

(8) Every student cheated.

- $\mathbf{every}(\mathbf{A}, \mathbf{B}) = 1$ iff $\mathbf{A} \subseteq \mathbf{B}$
- **student**: John, Mary, Sue
- **cheat**: John, Mary
- $\mathbf{student} \not\subseteq \mathbf{cheat} \Rightarrow \mathbf{every}(\mathbf{student}, \mathbf{cheat}) = 0$
- "Every student cheated" is false.

Binary Strings

- The language of **A** is the set of all permutations of **A**.

Example

student	John, Mary, Sue
L(**student**)	John Mary Sue, John Sue Mary
	Mary John Sue, Mary Sue John
	Sue John Mary, Sue Mary John

- Now replace every **a** $\in$ **A** by a truth value:

1	if **a** $\in$ **B**
0	if **a** $\notin$ **B**

- The result is the **binary string language** of **A** under **B**.

Example

student	John, Mary, Sue
cheat	John, Mary
binary strings	110, 101, 011

Binary Strings

- The language of **A** is the set of all permutations of **A**.

Example

student	John, Mary, Sue
$L($**student**$)$	John Mary Sue, John Sue Mary
	Mary John Sue, Mary Sue John
	Sue John Mary, Sue Mary John

- Now replace every **a** $\in$ **A** by a truth value:
  1 if **a** $\in$ **B**
  0 if **a** $\notin$ **B**
- The result is the **binary string language** of **A** under **B**.

Example

student	John, Mary, Sue
cheat	John, Mary
binary strings	110, 101, 011

Quantifier Languages (van Benthem 1986)

- We can associate each quantifier Q with a language in $\{0,1\}^*$ $\Rightarrow$ Q accepts only binary strings of specific shape
- This is its **quantifier language**.

Example: *every*

- **every**(**A**, **B**) holds iff **A** $\subseteq$ **B**
- So every element of **A** must be mapped to 1.
- $L($**every**$) = \{1\}^*$

Example: *some*

- **some**(**A**, **B**) holds iff **A** $\cap$ **B** $\neq \emptyset$
- Some element of **A** must be mapped to 1.
- $L($**some**$) = \{0,1\}^* \, 1 \, \{0,1\}^*$

Quantifier Languages (van Benthem 1986)

- We can associate each quantifier Q with a language in $\{0, 1\}^*$
  $\Rightarrow$ Q accepts only binary strings of specific shape
- This is its **quantifier language**.

Example: *every*

- **every**(**A**, **B**) holds iff **A** $\subseteq$ **B**
- So every element of **A** must be mapped to 1.
- $L(\textbf{every}) = \{1\}^*$

Example: *some*

- **some**(**A**, **B**) holds iff **A** $\cap$ **B** $\neq \emptyset$
- Some element of **A** must be mapped to 1.
- $L(\textbf{some}) = \{0, 1\}^* \, 1 \, \{0, 1\}^*$

Quantifier Languages (van Benthem 1986)

- We can associate each quantifier Q with a language in $\{0,1\}^*$
  $\Rightarrow$ Q accepts only binary strings of specific shape
- This is its **quantifier language**.

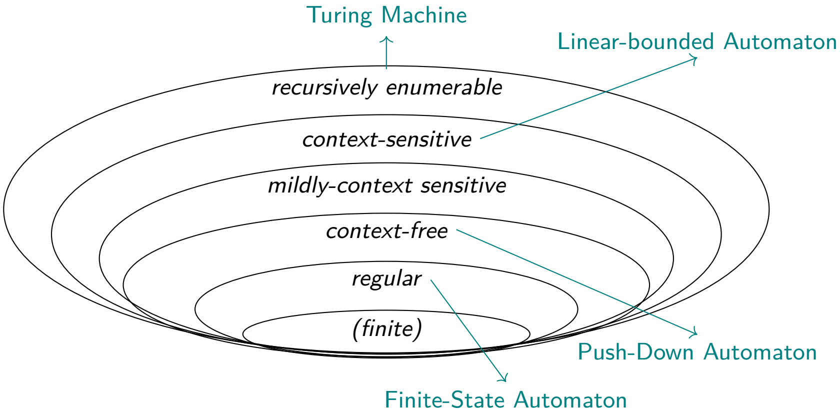
Example: *every*

- **every**(**A**, **B**) holds iff **A** $\subseteq$ **B**
- So every element of **A** must be mapped to 1.
- $L(\textbf{every}) = \{1\}^*$

Example: *some*

- **some**(**A**, **B**) holds iff **A** $\cap$ **B** $\neq \emptyset$
- Some element of **A** must be mapped to 1.
- $L(\textbf{some}) = \{0,1\}^* \, 1 \, \{0,1\}^*$

Chomsky Hierarchy and Automata Theory



Semantic Automata (van Benthem 1986, Mostowski 1998)

We can rank quantifiers based on their quantifier languages and the complexity of the machine needed to recognize them.

Aristotelian Quantifiers are FSA-recognizable

Reminder: *every*

- **every**(**A**, **B**) holds iff **A** $\subseteq$ **B**
- So every element of **A** must be mapped to 1.
- $L(\textbf{every}) = \{1\}^*$

False

student	John, Mary, Sue
cheat	John, Mary
binary strings	110, 101, 011

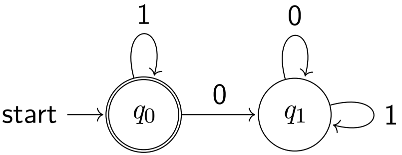
True

student	John, Mary, Sue
cheat	John, Mary,Sue
binary strings	111

Aristotelian Quantifiers are FSA-recognizable

Reminder: *every*

- **every**(**A**, **B**) holds iff **A** $\subseteq$ **B**
- So every element of **A** must be mapped to 1.
- $L($**every**$) = \{1\}^*$



False

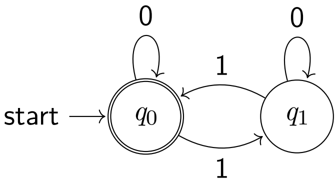
student	John, Mary, Sue
cheat	John, Mary
binary strings	110, 101, 011

True

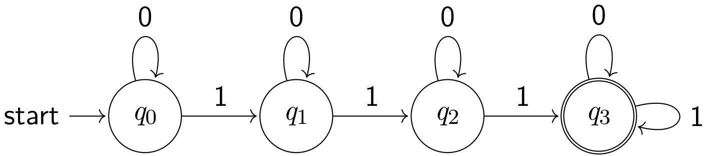
student	John, Mary, Sue
cheat	John, Mary,Sue
binary strings	111

Other FSA-recognizable quantifiers

- Parity quantifiers: **An even number**



- Cardinal quantifiers: **At least 3**



Proportional Quantifiers

- **most**(**A**, **B**) holds iff $|\mathbf{A} \cap \mathbf{B}| > |\mathbf{A} - \mathbf{B}|$
- $L_{most} := \{w \in \{0,1\}^* : |1|_w > |0|_w\}$
- There is no finite automaton recognizing this language.
- We need internal memory.
  $\Rightarrow$ **push-down automata**: two states + a stack

A Hierarchy of Quantifiers' Complexity

FSA		**PDA**
$\boxed{\{\textit{All, Some, Even, Odd, At least n, At most n}\}}$	$<$	$\boxed{\{\textit{Less than half, More than half, Most}\}}$

Are these all of equivalent complexity?

A Hierarchy of Quantifiers' Complexity

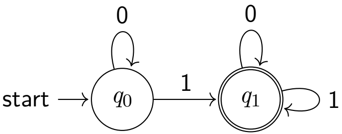
FSA **PDA**

$\boxed{\{\textit{All, Some, Even, Odd, At least n, At most n}\}} < \boxed{\{\textit{Less than half, More than half, Most}\}}$

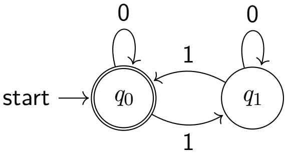
Are these all of equivalent complexity?

Let's Look at the Automata One More Time

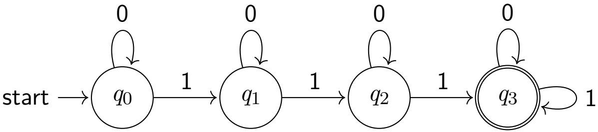
- Aristotelian quantifiers: **Some**



- Parity quantifiers: **An even number**



- Cardinal quantifiers: **At least 3**



A Hierarchy of Quantifiers' Complexity

FSA **PDA**

{*All, Some*} < {*Even, Odd*} < {*At least n, At most n*} < {*Less than half, More than half, Most*}

Are these all of equivalent complexity? (Szymanik 2016)

- Cyclic vs acyclic automata
- The number of states matters
- But: Complexity = succinctness of automata?

Reminder

It's all grounded in quantifier languages

- FSA recognizable quantifiers → Regular quantifier languages

A Hierarchy of Quantifiers' Complexity

FSA **PDA**

$\{$*All, Some*$\} < \{$*Even, Odd*$\} < \{$*At least n, At most n*$\} < \{$*Less than half, More than half, Most*$\}$

Are these all of equivalent complexity? (Szymanik 2016)

- Cyclic vs acyclic automata
- The number of states matters
- **But: Complexity = succinctness of automata?**

Reminder

It's all grounded in quantifier languages

- FSA recognizable quantifiers → Regular quantifier languages

A Hierarchy of Quantifiers' Complexity

FSA | **PDA**

$\{$*All, Some*$\} < \{$*Even, Odd*$\} < \{$*At least n, At most n*$\} < \{$*Less than half, More than half, Most*$\}$

Are these all of equivalent complexity? (Szymanik 2016)

- Cyclic vs acyclic automata
- The number of states matters
- **But: Complexity = succinctness of automata?**

Reminder

It's all grounded in quantifier languages

- FSA recognizable quantifiers → Regular quantifier languages

A Hierarchy of Quantifiers' Complexity

FSA		**PDA**
$\{$*All, Some*$\} < \{$*Even, Odd*$\} < \{$*At least n, At most n*$\}$	$<$	$\{$*Less than half, More than half, Most*$\}$

Are these all of equivalent complexity? (Szymanik 2016)

- Cyclic vs acyclic automata
- The number of states matters
- **But: Complexity = succinctness of automata?**

Reminder

It's all grounded in quantifier languages

- FSA recognizable quantifiers $\rightarrow$ Regular quantifier languages

Subregular Quantifiers: *Every is SL*

Reminder: *Every*

- **every**(**A**, **B**) holds iff **A** $\subseteq$ **B**
- $L(\textbf{every}) = \{1\}^*$
- Eg. *Every student cheated.*

False

student	John, Mary, Sue
cheat	John, Mary
binary strings	110, 101, 011
grammar	$*0$

True

student	John, Mary, Sue
cheat	John, Mary, Sue
binary strings	111
grammar	$*0$

Subregular Quantifiers: *Every is SL*

Reminder: *Every*

- **every**(**A**, **B**) holds iff **A** $\subseteq$ **B**
- $L($**every**$) = \{1\}^*$
- Eg. *Every student cheated.*

False

student	John, Mary, Sue
cheat	John, Mary
binary strings	110, 101, 011
grammar	^{*0}

True

student	John, Mary, Sue
cheat	John, Mary, Sue
binary strings	111
grammar	^{*0}

Subregular Quantifiers: *Every is SL*

Reminder: *Every*

- **every**(**A**, **B**) holds iff **A** $\subseteq$ **B**
- $L(\textbf{every}) = \{1\}^*$
- Eg. *Every student cheated.*

False

student	John, Mary, Sue
cheat	John, Mary
binary strings	110, 101, 011
grammar	^{*0}

$\rtimes$ 1 1 0 $\ltimes$

True

student	John, Mary, Sue
cheat	John, Mary, Sue
binary strings	111
grammar	^{*0}

$\rtimes$ 1 1 1 $\ltimes$

Subregular Quantifiers: *Every is SL*

Reminder: *Every*

- **every**(**A**, **B**) holds iff **A** $\subseteq$ **B**
- $L($**every**$) = \{1\}^*$
- Eg. *Every student cheated.*

False

student	John, Mary, Sue
cheat	John, Mary
binary strings	110, 101, 011
grammar	^{*0}

$\rtimes \; \boxed{1} \; 1 \; 0 \; \ltimes$

True

student	John, Mary, Sue
cheat	John, Mary, Sue
binary strings	111
grammar	^{*0}

$\rtimes \; 1 \; 1 \; 1 \; \ltimes$

Subregular Quantifiers: *Every is SL*

Reminder: *Every*

- **every**(**A**, **B**) holds iff **A** $\subseteq$ **B**
- $L($**every**$) = \{1\}^*$
- Eg. *Every student cheated.*

False

student	John, Mary, Sue
cheat	John, Mary
binary strings	110, 101, 011
grammar	^{*0}

$\rtimes$ 1 $\boxed{1}$ 0 $\ltimes$

True

student	John, Mary, Sue
cheat	John, Mary, Sue
binary strings	111
grammar	^{*0}

$\rtimes$ 1 1 1 $\ltimes$

Subregular Quantifiers: *Every is SL*

Reminder: *Every*

- **every**(**A**, **B**) holds iff **A** $\subseteq$ **B**
- $L(\textbf{every}) = \{1\}^*$
- Eg. *Every student cheated.*

False

student	John, Mary, Sue
cheat	John, Mary
binary strings	110, 101, 011
grammar	^{*0}

$\rtimes$ 1 1 $\boxed{0}$ $\ltimes$

True

student	John, Mary, Sue
cheat	John, Mary, Sue
binary strings	111
grammar	^{*0}

$\rtimes$ 1 1 1 $\ltimes$

Subregular Quantifiers: *Every is SL*

Reminder: *Every*

- **every**(**A**, **B**) holds iff **A** $\subseteq$ **B**
- L(**every**) $= \{1\}^*$
- Eg. *Every student cheated.*

False

student	John, Mary, Sue
cheat	John, Mary
binary strings	110, 101, 011
grammar	^{*0}

F

$\rtimes\ 1\ 1\ \boxed{0}\ \ltimes$

True

student	John, Mary, Sue
cheat	John, Mary, Sue
binary strings	111
grammar	^{*0}

$\rtimes\ 1\ 1\ 1\ \ltimes$

Subregular Quantifiers: *Every is SL*

Reminder: *Every*

- **every**(**A**, **B**) holds iff **A** $\subseteq$ **B**
- $L(\textbf{every}) = \{1\}^*$
- Eg. *Every student cheated.*

False

student	John, Mary, Sue
cheat	John, Mary
binary strings	110, 101, 011
grammar	^{*0}

F

$\rtimes$ 1 1 $\boxed{0}$ $\ltimes$

True

student	John, Mary, Sue
cheat	John, Mary, Sue
binary strings	111
grammar	^{*0}

$\rtimes$ $\boxed{1}$ 1 1 $\ltimes$

Subregular Quantifiers: *Every is SL*

Reminder: *Every*

- **every**(**A**, **B**) holds iff **A** $\subseteq$ **B**
- $L($**every**$) = \{1\}^*$
- Eg. *Every student cheated.*

False

student	John, Mary, Sue
cheat	John, Mary
binary strings	110, 101, 011
grammar	^{*0}

F

$\rtimes$ 1 1 0 $\ltimes$

True

student	John, Mary, Sue
cheat	John, Mary, Sue
binary strings	111
grammar	^{*0}

$\rtimes$ 1 $\boxed{1}$ 1 $\ltimes$

Subregular Quantifiers: *Every is SL*

Reminder: *Every*

- **every**(**A**, **B**) holds iff **A** $\subseteq$ **B**
- $L(\textbf{every}) = \{1\}^*$
- Eg. *Every student cheated.*

False

student	John, Mary, Sue
cheat	John, Mary
binary strings	110, 101, 011
grammar	^{*0}

F

$\rtimes \ 1 \ 1 \ \boxed{0} \ \ltimes$

True

student	John, Mary, Sue
cheat	John, Mary, Sue
binary strings	111
grammar	^{*0}

$\rtimes \ 1 \ 1 \ \boxed{1} \ \ltimes$

Subregular Quantifiers: *Every is SL*

Reminder: *Every*

- **every**(**A**, **B**) holds iff **A** $\subseteq$ **B**
- $L(\textbf{every}) = \{1\}^*$
- Eg. *Every student cheated.*

False

student	John, Mary, Sue
cheat	John, Mary
binary strings	110, 101, 011
grammar	^{*0}

F

$\rtimes$ 1 1 $\boxed{0}$ $\ltimes$

True

student	John, Mary, Sue
cheat	John, Mary, Sue
binary strings	111
grammar	^{*0}

T

$\rtimes$ 1 1 1 $\ltimes$

Subregular Quantifiers: *Some* is SL?

Reminder: *some*

- **some**(**A**, **B**) holds iff **A** $\cap$ **B** $\neq \emptyset$
- $L(\textbf{some}) = \{0,1\}^* \, 1 \, \{0,1\}^*$
- Eg. *Some student cheated.*

False

student	John, Mary, Sue
cheat	
binary strings	000
grammar	$*0$

True

student	John, Mary, Sue
cheat	John
binary strings	100,010,001
grammar	$*0$

Subregular Quantifiers: *Some* is SL?

Reminder: *some*

- **some**(**A**, **B**) holds iff **A** $\cap$ **B** $\neq \emptyset$
- $L(\textbf{some}) = \{0,1\}^* \, 1 \, \{0,1\}^*$
- Eg. *Some student cheated.*

False

student	John, Mary, Sue
cheat	
binary strings	000
grammar	^{*0}

True

student	John, Mary, Sue
cheat	John
binary strings	100,010,001
grammar	^{*0}

Subregular Quantifiers: *Some* is SL?

Reminder: *some*

- **some**(**A**, **B**) holds iff **A** $\cap$ **B** $\neq \emptyset$
- $L(\textbf{some}) = \{0,1\}^* \, 1 \, \{0,1\}^*$
- Eg. *Some student cheated.*

False

student	John, Mary, Sue
cheat	
binary strings	000
grammar	$*0$

$\rtimes$ 0 0 0 $\ltimes$

True

student	John, Mary, Sue
cheat	John
binary strings	100,010,001
grammar	$*0$

$\rtimes$ 0 0 1 $\ltimes$

Subregular Quantifiers: *Some* is SL?

Reminder: *some*

- **some**(**A**, **B**) holds iff $\mathbf{A} \cap \mathbf{B} \neq \emptyset$
- $L(\textbf{some}) = \{0,1\}^* \, 1 \, \{0,1\}^*$
- Eg. *Some student cheated.*

False

student	John, Mary, Sue
cheat	
binary strings	000
grammar	$*0$

$\rtimes \; \boxed{0} \; 0 \; 0 \; \ltimes$

True

student	John, Mary, Sue
cheat	John
binary strings	100,010,001
grammar	$*0$

$\rtimes \; 0 \; 0 \; 1 \; \ltimes$

Subregular Quantifiers: *Some* is SL?

Reminder: *some*

- **some**(**A**, **B**) holds iff $\mathbf{A} \cap \mathbf{B} \neq \emptyset$
- $L(\textbf{some}) = \{0,1\}^* \, 1 \, \{0,1\}^*$
- Eg. *Some student cheated.*

False

student	John, Mary, Sue
cheat	
binary strings	000
grammar	^{*0}

$\rtimes \ 0 \ \boxed{0} \ 0 \ \ltimes$

True

student	John, Mary, Sue
cheat	John
binary strings	100,010,001
grammar	^{*0}

$\rtimes \ 0 \ 0 \ 1 \ \ltimes$

Subregular Quantifiers: *Some* is SL?

Reminder: *some*

- $\mathbf{some}(\mathbf{A}, \mathbf{B})$ holds iff $\mathbf{A} \cap \mathbf{B} \neq \emptyset$
- $L(\mathbf{some}) = \{0,1\}^* \, 1 \, \{0,1\}^*$
- Eg. *Some student cheated.*

False

student	John, Mary, Sue
cheat	
binary strings	000
grammar	^{*0}

$\rtimes$ 0 0 $\boxed{0}$ $\ltimes$

True

student	John, Mary, Sue
cheat	John
binary strings	100,010,001
grammar	^{*0}

$\rtimes$ 0 0 1 $\ltimes$

Subregular Quantifiers: *Some* is SL?

Reminder: *some*

- **some**(**A**, **B**) holds iff **A** $\cap$ **B** $\neq \emptyset$
- $L($**some**$) = \{0,1\}^* \, 1 \, \{0,1\}^*$
- Eg. *Some student cheated.*

False

student	John, Mary, Sue
cheat	
binary strings	000
grammar	^{*0}

F

$\rtimes$ $\boxed{0}$ $\boxed{0}$ $\boxed{0}$ $\ltimes$

True

student	John, Mary, Sue
cheat	John
binary strings	100,010,001
grammar	^{*0}

$\rtimes$ 0 0 1 $\ltimes$

Subregular Quantifiers: *Some* is SL?

Reminder: *some*

- **some**(**A**, **B**) holds iff **A** $\cap$ **B** $\neq \emptyset$
- $L(\textbf{some}) = \{0,1\}^* \, 1 \, \{0,1\}^*$
- Eg. *Some student cheated.*

False

student	John, Mary, Sue
cheat	
binary strings	000
grammar	^{*0}

F

$\rtimes$ $\boxed{0}\boxed{0}\boxed{0}$ $\ltimes$

True

student	John, Mary, Sue
cheat	John
binary strings	100,010,001
grammar	^{*0}

$\rtimes$ $\boxed{0}\boxed{0}\boxed{1}$ $\ltimes$

Subregular Quantifiers: *Some* is SL?

Reminder: *some*

- **some**(**A**, **B**) holds iff **A** $\cap$ **B** $\neq \emptyset$
- $L(\textbf{some}) = \{0,1\}^* \, 1 \, \{0,1\}^*$
- Eg. *Some student cheated.*

False

student	John, Mary, Sue
cheat	
binary strings	000
grammar	^{*0}

F

$\rtimes$ $\boxed{0}\,\boxed{0}\,\boxed{0}$ $\ltimes$

True

student	John, Mary, Sue
cheat	John
binary strings	100,010,001
grammar	^{*0}

F

$\rtimes$ $\boxed{0}\,\boxed{0}\,\boxed{1}$ $\ltimes$

Subregular Quantifiers: *Some* is SL?

Reminder: *some*

- **some**(**A**, **B**) holds iff **A** $\cap$ **B** $\neq \emptyset$
- $L($**some**$) = \{0,1\}^* \, 1 \, \{0,1\}^*$
- Eg. *Some student cheated.*

False

student	John, Mary, Sue
cheat	
binary strings	000
grammar	^{*00}

F

$\rtimes$ 0 0 0 $\ltimes$

True

student	John, Mary, Sue
cheat	John
binary strings	100,010,001
grammar	^{*00}

F

$\rtimes$ 0 0 1 $\ltimes$

Subregular Quantifiers: *Some* is SL?

Reminder: *some*

- **some**(**A**, **B**) holds iff **A** $\cap$ **B** $\neq \emptyset$
- $L(\textbf{some}) = \{0,1\}^* \, 1 \, \{0,1\}^*$
- Eg. *Some student cheated.*

False

student	John, Mary, Sue
cheat	
binary strings	000
grammar	^{*000}

F

$\rtimes$ $\boxed{0\ 0\ 0}$ $\ltimes$

True

student	John, Mary, Sue
cheat	John
binary strings	100,010,001
grammar	^{*000}

T

$\rtimes$ $\boxed{0\ 0\ 1}$ $\ltimes$

Subregular Quantifiers: *Some* is SL?

Reminder: *some*

- **some**(**A**, **B**) holds iff **A** ∩ **B** $\neq \emptyset$
- $L(\textbf{some}) = \{0,1\}^* \, 1 \, \{0,1\}^*$
- Eg. *Some student cheated.*

False

student	John, Mary, Sue
cheat	
binary strings	000
grammar	??

$\rtimes \; \boxed{0 \; 0^n \; 0} \; \ltimes$

True

student	John, Mary, Sue
cheat	John
binary strings	100,010,001
grammar	??

$\rtimes \; \boxed{0 \; 0^n \; 1} \; \ltimes$

Subregular Quantifiers: *Some* is TSL

Reminder: *some*

- **some**(**A**, **B**) holds iff **A** $\cap$ **B** $\neq \emptyset$
- $L($**some**$) = \{0,1\}^* \, 1 \, \{0,1\}^*$
- Eg. *Some student cheated.*

False

student	John, Mary, Sue
cheat	
binary strings	000
grammar	$T = \{1\}$
	$S = \{{}^*\rtimes\ltimes\}$

$\rtimes$ 0 0 $\ltimes$

True

student	John, Mary, Sue
cheat	John,
binary strings	100, 010, 001
grammar	$T = \{1\}$
	$S = \{{}^*\rtimes\ltimes\}$

$\rtimes$ 1 $\ltimes$

Subregular Quantifiers: *Some* is TSL

Reminder: *some*

- **some**(**A**, **B**) holds iff $\mathbf{A} \cap \mathbf{B} \neq \emptyset$
- $L(\textbf{some}) = \{0,1\}^* \, 1 \, \{0,1\}^*$
- Eg. *Some student cheated.*

False

student	John, Mary, Sue
cheat	
binary strings	000
grammar	$T = \{1\}$
	$S = \{^*\rtimes\ltimes\}$

$\rtimes$ 0 0 $\ltimes$

True

student	John, Mary, Sue
cheat	John,
binary strings	100, 010, 001
grammar	$T = \{1\}$
	$S = \{^*\rtimes\ltimes\}$

$\rtimes$ 1 $\ltimes$

Subregular Quantifiers: *Some* is TSL

Reminder: *some*

- **some**(**A**, **B**) holds iff $\mathbf{A} \cap \mathbf{B} \neq \emptyset$
- $L(\textbf{some}) = \{0,1\}^* \, 1 \, \{0,1\}^*$
- Eg. *Some student cheated.*

False

student	John, Mary, Sue
cheat	
binary strings	000
grammar	$T = \{1\}$
	$S = \{{}^*\rtimes\ltimes\}$

$\rtimes$ 0 0 $\ltimes$

True

student	John, Mary, Sue
cheat	John,
binary strings	100, 010, 001
grammar	$T = \{1\}$
	$S = \{{}^*\rtimes\ltimes\}$

$\rtimes$ 1 $\ltimes$

Subregular Quantifiers: *Some* is TSL

Reminder: *some*

- **some**(**A**, **B**) holds iff $\mathbf{A} \cap \mathbf{B} \neq \emptyset$
- $L(\textbf{some}) = \{0, 1\}^* \, 1 \, \{0, 1\}^*$
- Eg. *Some student cheated.*

False

student	John, Mary, Sue
cheat	
binary strings	000
grammar	$T = \{1\}$
	$S = \{^* \rtimes \ltimes\}$

$\rtimes \quad 0 \quad 0 \quad 0 \quad \ltimes$

True

student	John, Mary, Sue
cheat	John,
binary strings	100, 010, 001
grammar	$T = \{1\}$
	$S = \{^* \rtimes \ltimes\}$

$\rtimes \quad 0 \quad 1 \quad 0 \quad \ltimes$

Subregular Quantifiers: *Some* is TSL

Reminder: *some*

- **some**(**A**, **B**) holds iff **A** $\cap$ **B** $\neq \emptyset$
- $L($**some**$) = \{0,1\}^* \, 1 \, \{0,1\}^*$
- Eg. *Some student cheated.*

False

student	John, Mary, Sue
cheat	
binary strings	000
grammar	$T = \{1\}$
	$S = \{^* \rtimes \ltimes\}$

$\rtimes$ $\ltimes$

$\rtimes \; 0 \; 0 \; 0 \; \ltimes$

True

student	John, Mary, Sue
cheat	John,
binary strings	100, 010, 001
grammar	$T = \{1\}$
	$S = \{^* \rtimes \ltimes\}$

$\rtimes \; 0 \; 1 \; 0 \; \ltimes$

Subregular Quantifiers: *Some* is TSL

Reminder: *some*

- **some**(**A**, **B**) holds iff $\mathbf{A} \cap \mathbf{B} \neq \emptyset$
- $L(\textbf{some}) = \{0,1\}^* \, 1 \, \{0,1\}^*$
- Eg. *Some student cheated.*

False

student	John, Mary, Sue
cheat	
binary strings	000
grammar	$T = \{1\}$
	$S = \{^* \rtimes \ltimes\}$

$\rtimes$ $\ltimes$

$\rtimes$ $\boxed{0}$ 0 0 $\ltimes$

True

student	John, Mary, Sue
cheat	John,
binary strings	100, 010, 001
grammar	$T = \{1\}$
	$S = \{^* \rtimes \ltimes\}$

$\rtimes$ 0 1 0 $\ltimes$

Subregular Quantifiers: *Some* is TSL

Reminder: *some*

- **some**(**A**, **B**) holds iff **A** $\cap$ **B** $\neq \emptyset$
- $L($**some**$) = \{0,1\}^* \, 1 \, \{0,1\}^*$
- Eg. *Some student cheated.*

False

student	John, Mary, Sue
cheat	
binary strings	000
grammar	$T = \{1\}$
	$S = \{^* \rtimes \ltimes\}$

$\rtimes$ $\ltimes$

$\rtimes \; 0 \; \boxed{0} \; 0 \; \ltimes$

True

student	John, Mary, Sue
cheat	John,
binary strings	100, 010, 001
grammar	$T = \{1\}$
	$S = \{^* \rtimes \ltimes\}$

$\rtimes \; 0 \; 1 \; 0 \; \ltimes$

Subregular Quantifiers: *Some* is TSL

Reminder: *some*

- **some**(**A**, **B**) holds iff **A** $\cap$ **B** $\neq \emptyset$
- $L($**some**$) = \{0,1\}^* \, 1 \, \{0,1\}^*$
- Eg. *Some student cheated.*

False

student	John, Mary, Sue
cheat	
binary strings	000
grammar	$T = \{1\}$
	$S = \{{}^*\rtimes\ltimes\}$

$\rtimes$ .... $\ltimes$

$\rtimes \; 0 \; 0 \; \boxed{0} \; \ltimes$

True

student	John, Mary, Sue
cheat	John,
binary strings	100, 010, 001
grammar	$T = \{1\}$
	$S = \{{}^*\rtimes\ltimes\}$

$\rtimes \; 0 \; 1 \; 0 \; \ltimes$

Subregular Quantifiers: *Some* is TSL

Reminder: *some*

- **some**(**A**, **B**) holds iff **A** $\cap$ **B** $\neq \emptyset$
- $L(\textbf{some}) = \{0,1\}^* \, 1 \, \{0,1\}^*$
- Eg. *Some student cheated.*

False

student	John, Mary, Sue
cheat	
binary strings	000
grammar	$T = \{1\}$
	$S = \{{}^*\rtimes\ltimes\}$

F $\boxed{\rtimes \quad \ltimes}$

$\rtimes \ 0 \ 0 \ 0 \ \ltimes$

True

student	John, Mary, Sue
cheat	John,
binary strings	100, 010, 001
grammar	$T = \{1\}$
	$S = \{{}^*\rtimes\ltimes\}$

$\rtimes \ 0 \ 1 \ 0 \ \ltimes$

Subregular Quantifiers: *Some* is TSL

Reminder: *some*

- **some**(**A**, **B**) holds iff **A** $\cap$ **B** $\neq \emptyset$
- $L(\textbf{some}) = \{0,1\}^* \, 1 \, \{0,1\}^*$
- Eg. *Some student cheated.*

False

student	John, Mary, Sue
cheat	
binary strings	000
grammar	$T = \{1\}$
	$S = \{^*\rtimes\ltimes\}$

F $\boxed{\rtimes \quad \ltimes}$

$\rtimes \ 0 \ 0 \ 0 \ \ltimes$

True

student	John, Mary, Sue
cheat	John,
binary strings	100, 010, 001
grammar	$T = \{1\}$
	$S = \{^*\rtimes\ltimes\}$

$\rtimes \quad \ltimes$

$\rtimes \ 0 \ 1 \ 0 \ \ltimes$

Subregular Quantifiers: *Some* is TSL

Reminder: *some*

- **some**(**A**, **B**) holds iff **A** $\cap$ **B** $\neq \emptyset$
- $L($**some**$) = \{0,1\}^* \, 1 \, \{0,1\}^*$
- Eg. *Some student cheated.*

False

student	John, Mary, Sue
cheat	
binary strings	000
grammar	$T = \{1\}$
	$S = \{^*\rtimes\ltimes\}$

$$\underbrace{\boxed{\rtimes \qquad \ltimes}}_{F}$$

$$\rtimes \; 0 \; 0 \; 0 \; \ltimes$$

True

student	John, Mary, Sue
cheat	John,
binary strings	100, 010, 001
grammar	$T = \{1\}$
	$S = \{^*\rtimes\ltimes\}$

$$\rtimes \qquad \ltimes$$

$$\rtimes \; \boxed{0} \; 1 \; 0 \; \ltimes$$

Subregular Quantifiers: *Some* is TSL

Reminder: *some*

- **some**(**A**, **B**) holds iff **A** $\cap$ **B** $\neq \emptyset$
- $L(\textbf{some}) = \{0,1\}^* \, 1 \, \{0,1\}^*$
- Eg. *Some student cheated.*

False

student	John, Mary, Sue
cheat	
binary strings	000
grammar	$T = \{1\}$
	$S = \{^*\rtimes\ltimes\}$

F $\rtimes \quad \ltimes$

$\rtimes \; 0 \; 0 \; 0 \; \ltimes$

True

student	John, Mary, Sue
cheat	John,
binary strings	100, 010, 001
grammar	$T = \{1\}$
	$S = \{^*\rtimes\ltimes\}$

$\rtimes \; 1 \; \ltimes$

$\rtimes \; 0 \; \boxed{1} \; 0 \; \ltimes$

Subregular Quantifiers: *Some* is TSL

Reminder: *some*

- **some**(**A**, **B**) holds iff $\mathbf{A} \cap \mathbf{B} \neq \emptyset$
- $L(\textbf{some}) = \{0, 1\}^* \, 1 \, \{0, 1\}^*$
- Eg. *Some student cheated.*

False

student	John, Mary, Sue
cheat	
binary strings	000
grammar	$T = \{1\}$
	$S = \{{}^*\rtimes\ltimes\}$

F $\boxed{\rtimes \quad \ltimes}$

$\rtimes \; 0 \; 0 \; 0 \; \ltimes$

True

student	John, Mary, Sue
cheat	John,
binary strings	100, 010, 001
grammar	$T = \{1\}$
	$S = \{{}^*\rtimes\ltimes\}$

$\rtimes \; 1 \; \ltimes$

$\rtimes \; 0 \; 1 \; \boxed{0} \; \ltimes$

Subregular Quantifiers: *Some* is TSL

Reminder: *some*

- **some**(**A**, **B**) holds iff **A** $\cap$ **B** $\neq \emptyset$
- $L(\textbf{some}) = \{0,1\}^* \, 1 \, \{0,1\}^*$
- Eg. *Some student cheated.*

False

student	John, Mary, Sue
cheat	
binary strings	000
grammar	$T = \{1\}$
	$S = \{{}^*\rtimes\ltimes\}$

$$F \ \boxed{\rtimes \qquad \ltimes}$$

$$\rtimes \ 0 \ 0 \ 0 \ \ltimes$$

True

student	John, Mary, Sue
cheat	John,
binary strings	100, 010, 001
grammar	$T = \{1\}$
	$S = \{{}^*\rtimes\ltimes\}$

$$\boxed{\rtimes \quad 1} \quad \ltimes$$

$$\rtimes \ 0 \ 1 \ 0 \ \ltimes$$

Subregular Quantifiers: *Some* is TSL

Reminder: *some*

- **some**(**A**, **B**) holds iff **A** $\cap$ **B** $\neq \emptyset$
- $L(\textbf{some}) = \{0,1\}^* \, 1 \, \{0,1\}^*$
- Eg. *Some student cheated.*

False

student	John, Mary, Sue
cheat	
binary strings	000
grammar	$T = \{1\}$
	$S = \{{}^*\rtimes\ltimes\}$

F $\boxed{\rtimes \quad \ltimes}$

$\rtimes \; 0 \; 0 \; 0 \; \ltimes$

True

student	John, Mary, Sue
cheat	John,
binary strings	100, 010, 001
grammar	$T = \{1\}$
	$S = \{{}^*\rtimes\ltimes\}$

$\boxed{\rtimes \quad 1} \; \boxed{1 \quad \ltimes}$

$\rtimes \; 0 \; 1 \; 0 \; \ltimes$

Subregular Quantifiers: *Some* is TSL

Reminder: *some*

- **some**(**A**, **B**) holds iff **A** $\cap$ **B** $\neq \emptyset$
- $L($**some**$) = \{0,1\}^* \, 1 \, \{0,1\}^*$
- Eg. *Some student cheated.*

False

student	John, Mary, Sue
cheat	
binary strings	000
grammar	$T = \{1\}$
	$S = \{^* \rtimes \ltimes\}$

F $\boxed{\rtimes \quad \ltimes}$

$\rtimes \; 0 \; 0 \; 0 \; \ltimes$

True

student	John, Mary, Sue
cheat	John,
binary strings	100, 010, 001
grammar	$T = \{1\}$
	$S = \{^* \rtimes \ltimes\}$

T $\boxed{\rtimes \quad 1} \; \boxed{1 \quad \ltimes}$

$\rtimes \; 0 \; 1 \; 0 \; \ltimes$

Subregular Quantifiers: *Some* is TSL

Reminder: *some*

- **some**(**A**, **B**) holds iff $\mathbf{A} \cap \mathbf{B} \neq \emptyset$
- $L(\textbf{some}) = \{0,1\}^* \, 1 \, \{0,1\}^*$
- Eg. *Some student cheated.*

False

student	John, Mary, Sue
cheat	
binary strings	000
grammar	$T = \{1\}$
	$S = \{{}^*\rtimes\ltimes\}$

F $\rtimes \quad \ltimes$

$\rtimes \;\; 0 \;\; 0^n \;\; 0 \;\; \ltimes$

True

student	John, Mary, Sue
cheat	John,
binary strings	100, 010, 001
grammar	$T = \{1\}$
	$S = \{{}^*\rtimes\ltimes\}$

T $\rtimes \quad 1 \quad \ltimes$

$\rtimes \;\; 0^n \;\; 1 \;\; 0^n \;\; \ltimes$

Parity Quantifiers?

An even number

- **An even number**(**A**, **B**) holds iff $|\mathbf{A} \cap \mathbf{B}| \geq 2n$, with $n > 0$
- $L(\textbf{even}) = \{w \in 0, 1^* s.t. |1|_w \geq 2n$, with $n > 0\}$

Is $L(\textbf{even})$ a TSL language?

F 1 1 1 0 0 T 1 1 1 1 0 F 1 1 1 1 1

Parity Quantifiers?

An even number

- **An even number**(**A**, **B**) holds iff $|$**A** $\cap$ **B**$| \geq 2n$, with $n > 0$
- $L($**even**$) = \{w \in 0, 1^* s.t. |1|_w \geq 2n$, with $n > 0\}$

Is $L($**even**$)$ a TSL language?

F 1 1 1 0 0   T 1 1 1 1 0   F 1 1 1 1 1

Parity Quantifiers?

An even number

- **An even number**(**A**, **B**) holds iff $|$**A** $\cap$ **B**$| \geq 2n$, with $n > 0$
- $L($**even**$) = \{w \in 0, 1^* s.t. |1|_w \geq 2n$, with $n > 0\}$

Is $L($**even**$)$ a TSL language?

F 1 1 1 0 0   T 1 1 1 1 0   F 1 1 1 1 1

Parity Quantifiers?

An even number

- **An even number**(**A**, **B**) holds iff $|$**A** $\cap$ **B**$| \geq 2n$, with $n > 0$
- $L($**even**$) = \{w \in 0, 1^* s.t. |1|_w \geq 2n$, with $n > 0\}$

Is $L($**even**$)$ a TSL language?

1 1 1	1 1 1 1	1 1 1 1 1
....	....	....
F 1 1 1 0 0	T 1 1 1 1 0	F 1 1 1 1 1

Parity Quantifiers?

An even number

- **An even number**(**A**, **B**) holds iff $|\mathbf{A} \cap \mathbf{B}| \geq 2n$, with $n > 0$
- $L(\textbf{even}) = \{w \in 0, 1^* s.t. |1|_w \geq 2n$, with $n > 0\}$

Is $L(\textbf{even})$ a TSL language?

F $\boxed{1\ 1\ 1}$ F $\boxed{1\ 1\ 1}$ 1 F $\boxed{1\ 1\ 1}$ 1 1

F 1 1 1 0 0 T 1 1 1 1 0 F 1 1 1 1 1

Parity Quantifiers?

An even number

- **An even number**(**A**, **B**) holds iff $|\mathbf{A} \cap \mathbf{B}| \geq 2n$, with $n > 0$
- $L(\textbf{even}) = \{w \in 0, 1^* s.t. |1|_w \geq 2n$, with $n > 0\}$

Is $L(\textbf{even})$ a TSL language?

F $\boxed{1 \ 1 \ 1}$	T $\boxed{1 \ 1 \ 1 \ 1}$	T $\boxed{1 \ 1 \ 1 \ 1}$ 1
F 1 1 1 0 0	T 1 1 1 1 0	F 1 1 1 1 1

Parity Quantifiers?

An even number

- **An even number**(**A**, **B**) holds iff $|\mathbf{A} \cap \mathbf{B}| \geq 2n$, with $n > 0$
- $L($**even**$) = \{w \in 0, 1^* s.t. |1|_w \geq 2n$, with $n > 0\}$

Is $L($**even**$)$ a TSL language?

F 1 1 1

F 1 1 1 0 0

T 1 1 1 1

T 1 1 1 1 0

F 1 1 1 1 1

F 1 1 1 1 1

Parity Quantifiers?

An even number

- **An even number**(**A**, **B**) holds iff $|\mathbf{A} \cap \mathbf{B}| \geq 2n$, with $n > 0$
- $L(\textbf{even}) = \{w \in 0, 1^* s.t. |1|_w \geq 2n, \text{ with } n > 0\}$

Is $L(\textbf{even})$ a TSL language?

F $\boxed{1\ 1\ 1}$	T $\boxed{1\ 1\ 1\ 1}$	F $\boxed{1\ 1\ 1\ 1\ 1}$
F 1 1 1 0 0	T 1 1 1 1 0	F 1 1 1 1 1

Since n is arbitrary, there is **no general TSL grammar** that can generate $L(\textbf{even})$.

Characterization of Quantifier Languages (Graf 2019)

Language	Constraint	Complexity	Subregular Grammar
every	$\lvert 0\rvert_w = 0$	SL-1	**S** $:= \{\neg 0\}$
no	$\lvert 1\rvert_w = 0$	SL-1	**S** $:= \{\neg 1\}$
some	$\lvert 1\rvert_w \geq 1$	TSL-2	**T** $:= \{1\}$, **S** $:= \{\neg \rtimes \ltimes\}$
not all	$\lvert 0\rvert_w \geq 1$	TSL-2	**T** $:= \{0\}$, **S** $:= \{\neg \rtimes \ltimes\}$
(at least) n	$\lvert 1\rvert_w \geq n$	TSL-$(n+1)$	**T** $:= \{1\}$, **S** $:= \left\{\neg \rtimes 1^k \ltimes\right\}_{k \leq n}$
(at most) n	$\lvert 1\rvert_w \leq n$	TSL-$(n+1)$	**T** $:= \{1\}$, **S** $:= \left\{\neg 1^{k+1}\right\}$
all but n	$\lvert 0\rvert_w = n$	TSL-$(n+1)$	**T** $:= \{0\}$, **S** $:= \left\{\neg 0^{n+1}, \neg \rtimes 0^k \ltimes\right\}_{k \leq n}$
even number	$\lvert 1\rvert_w = 2n,\ n \geq 0$	regular	**impossible**
most	$\lvert 1\rvert_w \geq \lvert 0\rvert_w$	context-free	**impossible**

A Complexity Hierarchy (Revisited)

- Semantic Automata predictions

FSA		PDA
$\{$*All, Some*$\} < \{$*Even, Odd*$\} < \{$*At least n, At most n*$\}$	$<$	$\{$*Less than half, More than half, Most*$\}$

- Subregular characterization predictions

SL		TSL		REG		CF
$\{$*All*$\}$	$<$	$\{$*Some, At least n, At most n*$\}$	$<$	$\{$*Even, Odd*$\}$	$<$	$\{$*Less than half, More than half, Most*$\}$

Automata vs Quantifier Languages

- complexity independent of the specific recognition machine
- what's the **cognitive reality** of these predictions?

A Complexity Hierarchy (Revisited)

- Semantic Automata predictions

FSA				PDA
{*All*, *Some*} < {*Even*, *Odd*} < {*At least n*, *At most n*}	<	{*Less than half*, *More than half*, *Most*}		

- Subregular characterization predictions

SL		TSL		REG		CF
{*All*}	<	{*Some*, *At least n*, *At most n*}	<	{*Even*, *Odd*}	<	{*Less than half*, *More than half*, *Most*}

Automata vs Quantifier Languages

- complexity independent of the specific recognition machine
- what's the **cognitive reality** of these predictions?

Mechanisms and Descriptive Models

Automata theoretic classes seem to presuppose [...] specific classes of recognition mechanisms, raising questions about whether these are necessarily relevant to the cognitive mechanisms under study.

Descriptive characterizations focus on the ***nature of the information*** *about the properties of a string (or structure) that is needed in order to distinguish those which exhibit a pattern from those which do not.*

What one can conclude is that whatever the actual mechanism is it must be sensitive to the kind of information that characterizes the descriptive class.

Rogers & Pullum 2011

Conclusion

- Many questions!
  - Laws underlying linguistics knowledge?
  - How complex are they?
  - Why are those the laws?
  - (How) are they reflected in behavior?
- Interplay of theory and data:
  - new typological claims
  - deeper understanding of formalism through data
  - new empirical questions
  - unification of diverse data points
  - direct ties to cognition/processing/learnability

Careful!

It's just another tool. We need to be **explicit** about the questions that we are asking and the connections we postulate!

Selected References I

Applegate, R.B. 1972. *Ineseno chumash grammar*. Doctoral Dissertation, University of California,Berkeley.

Avcu, Enes, and Arild Hestvik. 2020. Unlearnable phonotactics. *Glossa: a journal of general linguistics* 5.

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

De Santo, Aniello, Thomas Graf, and John E. Drury. 2017. Evaluating subregular distinctions in the complexity of generalized quantifiers. Talk at the ESSLLI Workshop on Quantifiers and Determiners (QUAD 2017), July 17 – 21, University of Toulouse, France.

Frey, Werner, and Hans-martin Gärtner. 2002. On the treatment of scrambling and adjunction in minimalist grammars. In *In Proceedings, Formal Grammar?02*. Citeseer.

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

Graf, Thomas. 2017. Why movement comes for free once you have adjunction. In *Proceedings of CLS 53*. URL `http://ling.auf.net/lingbuzz/003943`, (to appear).

Selected References II

Graf, Thomas. 2018. Why movement comes for free once you have adjunction. In *Proceedings of CLS 53*, ed. Daniel Edmiston, Marina Ermolaeva, Emre Hakgüder, Jackie Lai, Kathryn Montemurro, Brandon Rhodes, Amara Sankhagowit, and Miachel Tabatowski, 117–136.

Graf, Thomas. 2019. A subregular bound on the complexity of lexical quantifiers. In *Proceedings of the 22nd Amsterdam Colloquium*, ed. Julian J. Schlöder, Dean McHugh, and Floris Roelofsen, 455–464.

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

Hyman, Larry M. 2011. Tone: Is it different? *The Handbook of Phonological Theory, Second Edition* 197–239.

Kobele, Gregory M., Christian Retoré, and Sylvain Salvati. 2007. An automata-theoretic approach to Minimalism. In *Model Theoretic Syntax at 10*, ed. James Rogers and Stephan Kepser, 71–80.

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

Michaelis, Jens. 2004. Observations on strict derivational minimalism. *Electronic Notes in Theoretical Computer Science* 53:192–209.

Selected References III

Shafiei, Nazila, and Thomas Graf. 2019. The subregular complexity of syntactic islands. Ms., Stony Brook University.

Stabler, Edward P. 1997. Derivational Minimalism. In *Logical aspects of computational linguistics*, ed. Christian Retoré, volume 1328 of *Lecture Notes in Computer Science*, 68–95. Berlin: Springer.

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

Vu, Mai Ha, Nazila Shafiei, and Thomas Graf. 2019. Case assignment in TSL syntax: A case study. In *Proceedings of the Society for Computation in Linguistics (SCiL) 2019*, ed. Gaja Jarosz, Max Nelson, Brendan O'Connor, and Joe Pater, 267–276.

Of Black Swans and Flying Pigs

Of Black Swans and Flying Pigs

Of Black Swans and Flying Pigs

- Not a single data point, but classes of phenomena
- Value of restrictive theories: predictive and explanatory
- We learn from falsifying them too!

A Plethora of Testable Predictions

Observation

- Attested patterns **A** and **B** are TSL.
- But combined pattern **A**+**B** is not TSL.

Prediction

- **A**+**B** should be harder to learn than **A** and **B**

A Plethora of Testable Predictions

Observation

- Attested patterns **A** and **B** are TSL.
- But combined pattern **A**+**B** is not TSL.

Prediction

- **A**+**B** should be harder to learn than **A** and **B**

Morphotactics as Tier-Based Strictly Local Dependencies

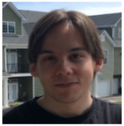
Alëna Aksënova Thomas Graf Sedigheh Moradi

Example: Compounding Markers

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

(9) vod **-o-** voz **-o-** voz
water -COMP- carry -COMP- carry
'carrier of water-carriers'

(10) türk bahçe kapı **-sı** (***-sı**)
turkish garden gate -COMP (*-COMP)
'Turkish garden gate'



Example: Compounding Markers [cont.]

- Russian and Turkish are TSL.

Tier$_1$	COMP affix and stem edges #
Russian n-grams	**oo**, **\$o**, **o\$**
Turkish n-grams	**sisi**, **\$si**, **si#**

- The combined pattern would yield **Ruskish**: stem^{n+1}-si^{n}
- This pattern is not regular and hence **not TSL either**.
- **Hypothesis (Aksenova et al, 2016)**
  If a language allows unboundedly many compound affixes, they are **infixes**.

Testable Predictions

- Can naive subjects learn Russian-like, Turkis-like, and Ruskish-like compounding?

Complexity as a Magnifying Lens

- We can compare patterns and predictions across classes
- We can also compare patterns within a same class

Proceedings of the Society for Computation in Linguistics

Volume 1 | Article 8

2018

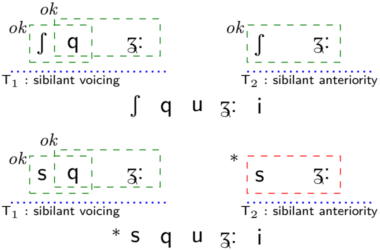
Formal Restrictions On Multiple Tiers

Alena Aksenova
Stony Brook University, alena.aksenova@stonybrook.edu

Sanket Deshmukh
Stony Brook University, sanket.deshmukh@stonybrook.edu

Testing Harmony Systems

- We can also account for multiple processes
- Thus we can cover the complete phonotactics of a language



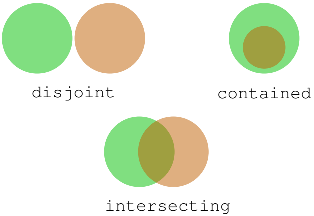


Figure 2: Theoretically possible tier alphabet relations

Testing Harmony Systems (cont.)

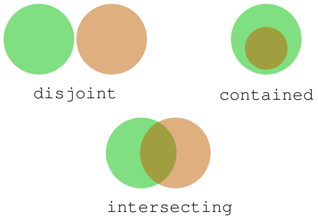
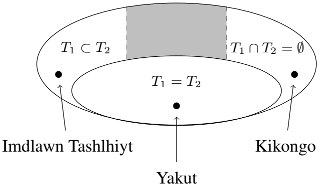


Figure 2: Theoretically possible tier alphabet relations



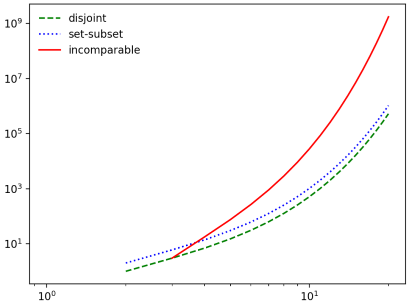
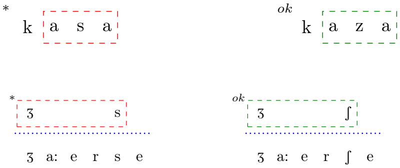


Figure 7: Growth of number of partitions of sets containing up to 20 elements (loglog scale)

The Fallacy of Generalization

- Imagine we want to test the ability to learn long-distance dependencies:



- Assuming an alphabet $\Sigma = \{a, b, c, d, e\}$, the training samples could look like the following:

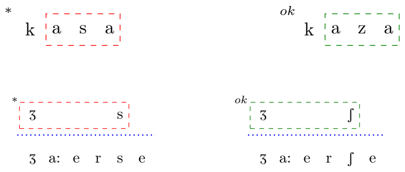
$$L_{loc} = \{abcd, aabcd, baacd, bcaae, \ldots\}$$
$$L_{dist} = \{abacd, bacad, bcada, bcaea, \ldots\}$$

What happens if we test on stimuli with similar distances?

$$L_{test} = \{abcad, abcad, bacda, abcea, \ldots\}$$

The Fallacy of Generalization

- Imagine we want to test the ability to learn long-distance dependencies:



- Assuming an alphabet $\Sigma = \{a, b, c, d, e\}$, the training samples could look like the following:

$$L_{loc} = \{abcd, aabcd, baacd, bcaae, \dots\}$$
$$L_{dist} = \{abacd, bacad, bcada, bcaea, \dots\}$$

What happens if we test on stimuli with similar distances?

$$L_{test} = \{abcad, abcad, bacda, abcea, \dots\}$$