Subregular Morpho-Semantics

The Expressive Limits of Monomorphemic Quantifiers

Thomas Graf

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

You can get the slides here under "News"

Rutgers University
December 15, 2017

Take-Home Message

- Supplement linguistic theory with computational perspective
- Typological gaps can be explained computationally.

Case Study: Morphosemantics of Quantifiers

A D-quantifier may have a monomorphemic realization
only if its quantifier language is TSL.

Outline

Subregular Hierarchy

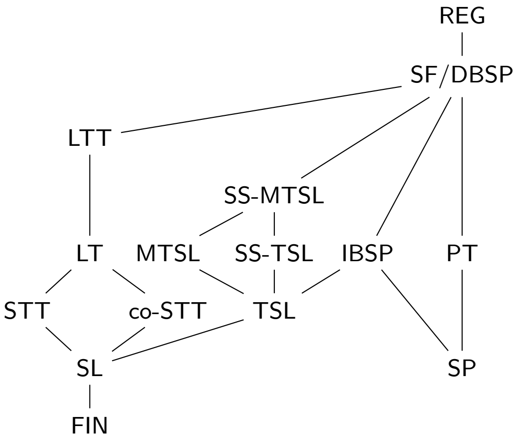
- **Subregular** hierarchy as measuring rod for complexity (Heinz 2009, 2010; Heinz et al. 2011; Chandlee 2014; Jardine 2016; McMullin 2016; Graf 2017)

1. define different classes of grammars
2. organize these classes into an expressivity hierarchy
3. needed level of expressivity?

Subregular Hierarchy

- **Subregular** hierarchy as measuring rod for complexity (Heinz 2009, 2010; Heinz et al. 2011; Chandlee 2014; Jardine 2016; McMullin 2016; Graf 2017)

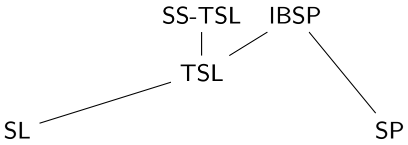
1. define different classes of grammars
2. organize these classes into an expressivity hierarchy
3. needed level of expressivity?



Subregular Hierarchy

- **Subregular** hierarchy as measuring rod for complexity (Heinz 2009, 2010; Heinz et al. 2011; Chandlee 2014; Jardine 2016; McMullin 2016; Graf 2017)

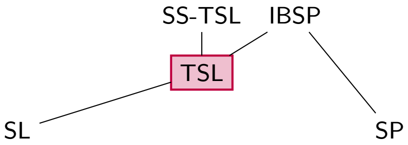
1. define different classes of grammars
2. organize these classes into an expressivity hierarchy
3. needed level of expressivity?



Subregular Hierarchy

- **Subregular** hierarchy as measuring rod for complexity (Heinz 2009, 2010; Heinz et al. 2011; Chandlee 2014; Jardine 2016; McMullin 2016; Graf 2017)

1. define different classes of grammars
2. organize these classes into an expressivity hierarchy
3. needed level of expressivity?



TSL: Tier-Based Strictly Local

- All patterns described by markedness constraints that are
  - inviolable,
  - locally bounded,
  - formalized as n-grams.
- Non-local dependencies are **local over tiers**.
  (Goldsmith 1976)
- **Linguistic core idea:**
  Dependencies are local over the right structure.

Example Without Tier: Word-Final Devoicing

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

Example: German

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

Example Without Tier: Word-Final Devoicing

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

Example: German

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

Example Without Tier: Word-Final Devoicing

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

Example: German

* $ r a d $

*z$

*v$

*d$

Example Without Tier: Word-Final Devoicing

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

Example: German

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

Example Without Tier: Word-Final Devoicing

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

Example: German

* $ r a d $

*z$

*v$

*d$

Example Without Tier: Word-Final Devoicing

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

Example: German

* $ r a d $

*z$

*v$

*d$

Example Without Tier: Word-Final Devoicing

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

Example: German

* $ r a d $ → *d$

*z$
*v$
*d$

Example Without Tier: Word-Final Devoicing

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

Example: German

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

Example Without Tier: Word-Final Devoicing

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

Example: German

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

Example Without Tier: Word-Final Devoicing

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

Example: German

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

Example Without Tier: Word-Final Devoicing

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

Example: German

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

Example Without Tier: Word-Final Devoicing

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

Example: German

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

Example Without Tier: Intervocalic Voicing

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

Example

* $ a z u s a $

Example Without Tier: Intervocalic Voicing

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

Example

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

Example Without Tier: Intervocalic Voicing

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

Example

*$ a z u s a $

Example Without Tier: Intervocalic Voicing

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

Example

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

Example Without Tier: Intervocalic Voicing

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

Example

* $ a z u s a $

Example Without Tier: Intervocalic Voicing

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

Example

* $ a z u s a $

Adding Tiers: Samala Sibilant Harmony

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

*sʃ *sʒ *zʃ *zʒ
*ʃs *ʒs *ʃz *ʒz

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

Example: Samala

*$ h a s x i n t i l a w a ʃ $

$ h a ʃ x i n t i l a w a ʃ $

Adding Tiers: Samala Sibilant Harmony

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

*sʃ *sʒ *zʃ *zʒ
*ʃs *ʒs *ʃz *ʒz

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

Example: Samala

*$ h a s x i n t i l a w a ʃ $

$ h a ʃ x i n t i l a w a ʃ $

Adding Tiers: Samala Sibilant Harmony

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

*sʃ *sʒ *zʃ *zʒ
*ʃs *ʒs *ʃz *ʒz

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

Example: Samala

*$ h a s x i n t i l a w a ʃ $

$ h a ʃ x i n t i l a w a ʃ $

Adding Tiers: Samala Sibilant Harmony

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

  *sʃ *sʒ *zʃ *zʒ
  *ʃs *ʒs *ʃz *ʒz

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

Example: Samala

*$ h a s x i n t i l a w a ʃ $

$ h a ʃ x i n t i l a w a ʃ $

Adding Tiers: Samala Sibilant Harmony

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

  *sʃ *sʒ *zʃ *zʒ
  *ʃs *ʒs *ʃz *ʒz

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

Example: Samala

*$ h a s x i n t i l a w a ʃ $

$ h a ʃ x i n t i l a w a ʃ $

*$ s t a j a n o w o n w a ʃ $

Adding Tiers: Samala Sibilant Harmony

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

*sʃ	*sʒ	*zʃ	*zʒ
*ʃs	*ʒs	*ʃz	*ʒz

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

Example: Samala

*$ h a s x i n t i l a w a ʃ $

$ h a ʃ x i n t i l a w a ʃ $

*$ s t a j a n o w o n w a ʃ $

Making Long-Distance Dependencies Local

- Let's take a hint from phonology: create locality with a **tier**. (Heinz et al. 2011)
- **Restriction 1:** only 1 tier
- **Restriction 2:** projection is determined by the segments, not their environment

Jeff Heinz

Example: Samala Revisited

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

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

Making Long-Distance Dependencies Local

- Let's take a hint from phonology: create locality with a **tier**. (Heinz et al. 2011)
- **Restriction 1:** only 1 tier
- **Restriction 2:** projection is determined by the segments, not their environment

Jeff Heinz

Example: Samala Revisited

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

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

Making Long-Distance Dependencies Local

- Let's take a hint from phonology: create locality with a **tier**. (Heinz et al. 2011)
- **Restriction 1:** only 1 tier
- **Restriction 2:** projection is determined by the segments, not their environment

Jeff Heinz

Example: Samala Revisited

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

Tier: $ s ʃ $

*$ h a s x i n t i l a w a ʃ $

$ h a ʃ x i n t i l a w a ʃ $

Making Long-Distance Dependencies Local

- Let's take a hint from phonology: create locality with a **tier**. (Heinz et al. 2011)
- **Restriction 1:** only 1 tier
- **Restriction 2:** projection is determined by the segments, not their environment

Jeff Heinz

Example: Samala Revisited

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

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

$ ʃ ʃ $
$ h a ʃ x i n t i l a w a ʃ $

Making Long-Distance Dependencies Local

- Let's take a hint from phonology: create locality with a **tier**. (Heinz et al. 2011)
- **Restriction 1:** only 1 tier
- **Restriction 2:** projection is determined by the segments, not their environment

Jeff Heinz

Example: Samala Revisited

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

$ s ʃ $ $ ʃ ʃ $

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

Making Long-Distance Dependencies Local

- Let's take a hint from phonology: create locality with a **tier**. (Heinz et al. 2011)
- **Restriction 1:** only 1 tier
- **Restriction 2:** projection is determined by the segments, not their environment

Jeff Heinz

Example: Samala Revisited

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

$ s ʃ $

*$ h a s x i n t i l a w a ʃ $

$ ʃ ʃ $

$ h a ʃ x i n t i l a w a ʃ $

Making Long-Distance Dependencies Local

- Let's take a hint from phonology: create locality with a **tier**. (Heinz et al. 2011)
- **Restriction 1:** only 1 tier
- **Restriction 2:** projection is determined by the segments, not their environment

Jeff Heinz

Example: Samala Revisited

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

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

Making Long-Distance Dependencies Local

- Let's take a hint from phonology: create locality with a **tier**. (Heinz et al. 2011)
- **Restriction 1:** only 1 tier
- **Restriction 2:** projection is determined by the segments, not their environment

Jeff Heinz

Example: Samala Revisited

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

$ s ʃ $ $ ʃ ʃ $

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

Making Long-Distance Dependencies Local

- Let's take a hint from phonology: create locality with a **tier**. (Heinz et al. 2011)
- **Restriction 1:** only 1 tier
- **Restriction 2:** projection is determined by the segments, not their environment

Jeff Heinz

Example: Samala Revisited

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

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

Culminativity: Simple Counting with TSL

Culminativity phonological word contains exactly 1 stress

Example

1. Project stress tier
2. *$$, *V́V́

*$ b a n á n a $ $ b á n á n a $ $ b a n a n a $

Culminativity: Simple Counting with TSL

Culminativity phonological word contains exactly 1 stress

Example

1. Project stress tier
2. *$$, *ÝÝ

*$ b a n á n a $ $ b á n á n a $ $ b a n a n a $

Culminativity: Simple Counting with TSL

Culminativity phonological word contains exactly 1 stress

Example

1. Project stress tier
2. *$$, *V́V́

```
  $      á    $
  |      |    |
* $ b a n á n a $        $ b á n á n a $        $ b a n a n a $
```

Culminativity: Simple Counting with TSL

Culminativity phonological word contains exactly 1 stress

Example

1. Project stress tier
2. *$$, *V́V́

```
  $     á   $          $  á  á   $
  |     |   |          |  |  |   |
* $ b a n á n a $      $ b á n á n a $      $ b a n a n a $
```

Culminativity: Simple Counting with TSL

Culminativity phonological word contains exactly 1 stress

Example

1. Project stress tier
2. *$$, * V́V́

$ á $ — *$ b a n á n a $

$ á á $ — $ b á n á n a $

$ $ — $ b a n a n a $

Why is TSL Interesting?

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

Example: First-Last Harmony

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

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

Why is TSL Interesting?

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

Example: First-Last Harmony

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

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

Why is TSL Interesting?

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

Example: First-Last Harmony

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

$ s ʃ $

$ h a s x i n t i l a w a ʃ $

*$ s t a j a n o w o n w a ʃ $

Why is TSL Interesting?

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

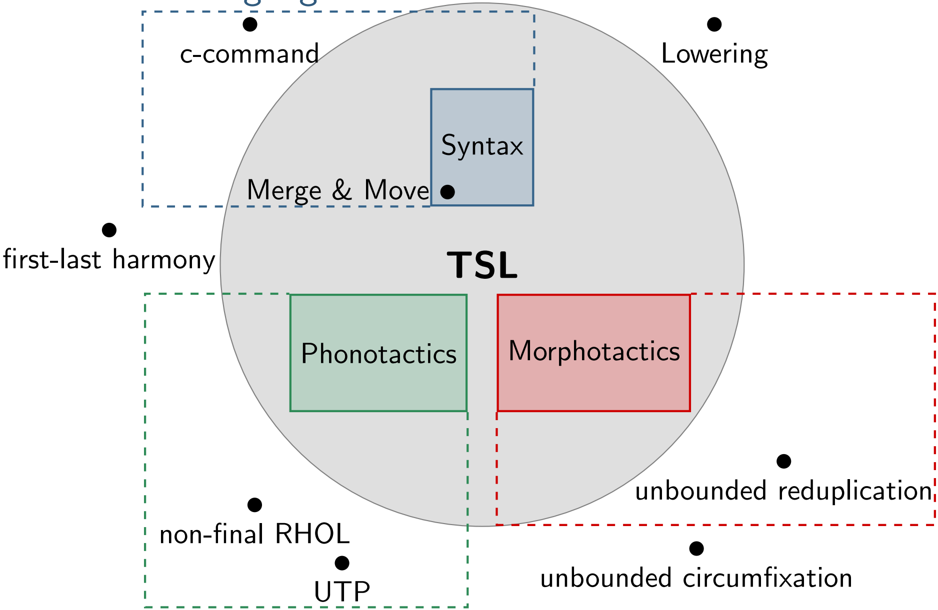
Example: First-Last Harmony

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

$ s ʃ $ $ s ʃ $

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

TSL Across Language Modules



TSL Semantics

- TSL seems to play an important role in
  - phonology,
  - morphology,
  - syntax.
- What's missing? Semantics!
- But TSL is about strings/trees.
- What is a **semantic string language**?

Formal Language Theory for Semantics

1. **Quantifier Languages**
   Meanings as strings of truth values
   (van Benthem 1986)
2. **“Tense Languages”**
   Meanings as strings of events
   (Fernando 2011)

- I’ll only talk about quantifier languages here.
- Ongoing work with **Rob Pasternak** on subregularity of tense languages

Evaluating the Truth of Quantifiers

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

students	John	Mary	Sue
cheated	yes	no	yes
string	Y	N	Y

- (1a): **False**, because the string contains a N
- (1b): **False**, because the string contains a Y
- (1c): **True**, because the string contains a Y
- (1d): **False**, because the string does not contain three Ys

Evaluating the Truth of Quantifiers

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

students	John	Mary	Sue
cheated	yes	no	yes
string	Y	N	Y

- (1a): **False**, because the string contains a N
- (1b): **False**, because the string contains a Y
- (1c): **True**, because the string contains a Y
- (1d): **False**, because the string does not contain three Ys

Formalization Step 1: Binary String Languages

Idea: Convert relation between sets **A** and **B** into set of **Yes/No-strings**

Definition (Binary String Language)

1. **A**, **B**: arbitrary sets
2. f(**A**, **B**): maps each **a** $\in$ **A** to Y if **a** $\in$ **B**, otherwise N
3. e(**A**): arbitrary enumeration of **A**
4. **L**(**A**, **B**): all e(**A**), relabeled by f(**A**, **B**)

Example

1. **Set of students**: {John, Mary, Sue}
   Set of cheaters: {John, Sue, Bill, Peter}

2. **f**(**A**, **B**):

$$\begin{array}{rcl} \text{John} & \mapsto & \text{Y} \\ \text{Mary} & \mapsto & \text{N} \\ \text{Sue} & \mapsto & \text{Y} \end{array}$$

3. **e**(**A**):

1)	John	Mary	Sue
2)	John	Sue	Mary
3)	Mary	John	Sue
4)	Mary	Sue	John
5)	Sue	John	Mary
6)	Sue	Mary	John

4. **L**(**A**, **B**):

$$\left\{ \begin{array}{c} \text{YNY}, \\ \text{YYN}, \\ \text{NYY} \end{array} \right\}$$

Formalization Step 2: Quantifier Language

Idea: Every quantifier is a set of acceptable **Yes/No-strings**

Definition (Quantifier Language)

$\mathbf{L}(\mathbf{Q})$ is the **quantifier language** of $\mathbf{Q}$ iff it holds for all $\mathbf{A}$ and $\mathbf{B}$ that $\mathbf{Q}(\mathbf{A}, \mathbf{B})$ is true iff $\mathbf{L}(\mathbf{A}, \mathbf{B}) \subseteq \mathbf{L}(\mathbf{Q})$.

Example

- $\mathbf{L}$(**every**) = set of all strings containing no N
- Why?
  - **every**($\mathbf{A}$, $\mathbf{B}$) iff $\mathbf{A} \subseteq \mathbf{B}$
  - If $\mathbf{A} \subseteq \mathbf{B}$, then no binary string contains N.
  - If some binary string contains N, then $\mathbf{A} \not\subseteq \mathbf{B}$.

Formalization Step 2: Quantifier Language

Idea: Every quantifier is a set of acceptable **Yes/No-strings**

Definition (Quantifier Language)

$\mathbf{L}(\mathbf{Q})$ is the **quantifier language** of $\mathbf{Q}$ iff it holds for all $\mathbf{A}$ and $\mathbf{B}$ that $\mathbf{Q}(\mathbf{A}, \mathbf{B})$ is true iff $\mathbf{L}(\mathbf{A}, \mathbf{B}) \subseteq \mathbf{L}(\mathbf{Q})$.

Example

- $\mathbf{L}(\textbf{every})$ = set of all strings containing no N
- Why?
  - **every**($\mathbf{A}$, $\mathbf{B}$) iff $\mathbf{A} \subseteq \mathbf{B}$
  - If $\mathbf{A} \subseteq \mathbf{B}$, then no binary string contains N.
  - If some binary string contains N, then $\mathbf{A} \not\subseteq \mathbf{B}$.

A Sample of Quantifier Languages

Quantifier	Constraint
every	
no	
some	
at least **n**	
at most **n**	
exactly **n**	
not all	
all but **n**	
most	
an even number	

A Sample of Quantifier Languages

Quantifier	Constraint
every	$\lvert \mathbf{N} \rvert = 0$
no	
some	
at least **n**	
at most **n**	
exactly **n**	
not all	
all but **n**	
most	
an even number	

A Sample of Quantifier Languages

Quantifier	Constraint
every	$\lvert \mathbf{N} \rvert = 0$
no	$\lvert \mathbf{Y} \rvert = 0$
some	
at least **n**	
at most **n**	
exactly **n**	
not all	
all but **n**	
most	
an even number	

A Sample of Quantifier Languages

Quantifier	Constraint
every	$\lvert \mathbf{N} \rvert = 0$
no	$\lvert \mathbf{Y} \rvert = 0$
some	$\lvert \mathbf{Y} \rvert \geq 1$
at least **n**	
at most **n**	
exactly **n**	
not all	
all but **n**	
most	
an even number	

A Sample of Quantifier Languages

Quantifier	Constraint
every	$\lvert \mathbf{N} \rvert = 0$
no	$\lvert \mathbf{Y} \rvert = 0$
some	$\lvert \mathbf{Y} \rvert \geq 1$
at least **n**	$\lvert \mathbf{Y} \rvert \geq \mathbf{n}$
at most **n**	
exactly **n**	
not all	
all but **n**	
most	
an even number	

A Sample of Quantifier Languages

Quantifier	Constraint
every	$\|\mathbf{N}\| = 0$
no	$\|\mathbf{Y}\| = 0$
some	$\|\mathbf{Y}\| \geq 1$
at least **n**	$\|\mathbf{Y}\| \geq \mathbf{n}$
at most **n**	$\|\mathbf{Y}\| \leq \mathbf{n}$
exactly **n**	
not all	
all but **n**	
most	
an even number	

A Sample of Quantifier Languages

Quantifier	Constraint
every	$\lvert \mathbf{N} \rvert = 0$
no	$\lvert \mathbf{Y} \rvert = 0$
some	$\lvert \mathbf{Y} \rvert \geq 1$
at least **n**	$\lvert \mathbf{Y} \rvert \geq \mathbf{n}$
at most **n**	$\lvert \mathbf{Y} \rvert \leq \mathbf{n}$
exactly **n**	$\lvert \mathbf{Y} \rvert = \mathbf{n}$
not all	
all but **n**	
most	
an even number	

A Sample of Quantifier Languages

Quantifier	**Constraint**
every	$\lvert \mathbf{N} \rvert = 0$
no	$\lvert \mathbf{Y} \rvert = 0$
some	$\lvert \mathbf{Y} \rvert \geq 1$
at least **n**	$\lvert \mathbf{Y} \rvert \geq \mathbf{n}$
at most **n**	$\lvert \mathbf{Y} \rvert \leq \mathbf{n}$
exactly **n**	$\lvert \mathbf{Y} \rvert = \mathbf{n}$
not all	$\lvert \mathbf{N} \rvert \geq 1$
all but **n**	
most	
an even number	

A Sample of Quantifier Languages

Quantifier	Constraint
every	$\lvert \mathbf{N} \rvert = 0$
no	$\lvert \mathbf{Y} \rvert = 0$
some	$\lvert \mathbf{Y} \rvert \geq 1$
at least **n**	$\lvert \mathbf{Y} \rvert \geq \mathbf{n}$
at most **n**	$\lvert \mathbf{Y} \rvert \leq \mathbf{n}$
exactly **n**	$\lvert \mathbf{Y} \rvert = \mathbf{n}$
not all	$\lvert \mathbf{N} \rvert \geq 1$
all but **n**	$\lvert \mathbf{N} \rvert = \mathbf{n}$
most	
an even number	

A Sample of Quantifier Languages

Quantifier	Constraint
every	$\lvert \mathbf{N} \rvert = 0$
no	$\lvert \mathbf{Y} \rvert = 0$
some	$\lvert \mathbf{Y} \rvert \geq 1$
at least **n**	$\lvert \mathbf{Y} \rvert \geq \mathbf{n}$
at most **n**	$\lvert \mathbf{Y} \rvert \leq \mathbf{n}$
exactly **n**	$\lvert \mathbf{Y} \rvert = \mathbf{n}$
not all	$\lvert \mathbf{N} \rvert \geq 1$
all but **n**	$\lvert \mathbf{N} \rvert = \mathbf{n}$
most	$\lvert \mathbf{Y} \rvert > \lvert \mathbf{N} \rvert$
an even number	

A Sample of Quantifier Languages

Quantifier	**Constraint**
every	$\|N\| = 0$
no	$\|Y\| = 0$
some	$\|Y\| \geq 1$
at least **n**	$\|Y\| \geq n$
at most **n**	$\|Y\| \leq n$
exactly **n**	$\|Y\| = n$
not all	$\|N\| \geq 1$
all but **n**	$\|N\| = n$
most	$\|Y\| > \|N\|$
an even number	$\|Y\|$ even

TSL Quantifier Languages for *every* and *no*

every is TSL Without Tiers

*N

$ Y Y Y Y $ $ Y Y N Y $

no is TSL Without Tiers

*Y

$ N N N N $ $ N N Y N $

TSL Quantifier Languages for *every* and *no*

every is TSL Without Tiers

*N

$ Y Y Y Y $ $ Y Y N Y $

no is TSL Without Tiers

*Y

$ N N N N $ $ N N Y N $

TSL Quantifier Languages for *every* and *no*

every is TSL Without Tiers

*N

$ Y Y Y Y $ $ Y Y N Y $

no is TSL Without Tiers

*Y

$ N N N N $ $ N N Y N $

TSL Quantifier Languages for *every* and *no*

every is TSL Without Tiers

*N

$ Y Y Y Y $ $ Y Y N Y $

no is TSL Without Tiers

*Y

$ N N N N $ $ N N Y N $

TSL Quantifier Languages for *every* and *no*

every is TSL Without Tiers

*N

$ Y Y Y Y $ $ Y Y N Y $

no is TSL Without Tiers

*Y

$ N N N N $ $ N N Y N $

TSL Quantifier Languages for *every* and *no*

every is TSL Without Tiers

*N

$ Y Y Y Y $ $ Y Y N Y $

no is TSL Without Tiers

*Y

$ N N N N $ $ N N Y N $

TSL Quantifier Languages for *every* and *no*

every is TSL Without Tiers

*N

$ Y Y Y Y $ $ Y Y N Y $

no is TSL Without Tiers

*Y

$ N N N N $ $ N N Y N $

TSL Quantifier Languages for *every* and *no*

every is TSL Without Tiers

*N

$ Y Y Y Y $ $ Y Y N Y $

no is TSL Without Tiers

*Y

$ N N N N $ $ N N Y N $

TSL Quantifier Languages for *every* and *no*

every is TSL Without Tiers

*N

$ Y Y Y Y $ $ Y Y N Y $

no is TSL Without Tiers

*Y

$ N N N N $ $ N N Y N $

TSL Quantifier Languages for *every* and *no*

every is TSL Without Tiers

*N

$ Y Y Y Y $ $ Y Y N Y $

no is TSL Without Tiers

*Y

$ N N N N $ $ N N Y N $

TSL Quantifier Languages for *every* and *no*

every is TSL Without Tiers

*N

$ Y Y Y Y $ $ Y Y N Y $

no is TSL Without Tiers

*Y

$ N N N N $ $ N N Y N $

TSL Quantifier Languages for *every* and *no*

every is TSL Without Tiers

*N

$ Y Y Y Y $ $ Y Y N Y $

no is TSL Without Tiers

*Y

$ N N N N $ $ N N Y N $

TSL Quantifier Languages for *every* and *no*

every is TSL Without Tiers

*N

$ Y Y Y Y $ $ Y Y N Y $

no is TSL Without Tiers

*Y

$ N N N N $ $ N N Y N $

TSL Quantifier Languages for *every* and *no*

every is TSL Without Tiers

*N

$ Y Y Y Y $ $ Y Y N Y $

no is TSL Without Tiers

*Y

$ N N N N $ $ N N Y N $

TSL Quantifier Languages for *every* and *no*

every is TSL Without Tiers

*N

$ Y Y Y Y $ $ Y Y N Y $

no is TSL Without Tiers

*Y

$ N N N N $ $ N N Y N $

TSL Quantifier Languages for *every* and *no*

every is TSL Without Tiers

*N

$ Y Y Y Y $ $ Y Y N Y $

no is TSL Without Tiers

*Y

$ N N N N $ $ N N Y N $

TSL Quantifier Languages for *every* and *no*

every is TSL Without Tiers

*N

$ Y Y Y Y $ $ Y Y N Y $

no is TSL Without Tiers

*Y

$ N N N N $ $ N N Y N $

TSL Quantifier Languages for *every* and *no*

every is TSL Without Tiers

*N

$ Y Y Y Y $ $ Y Y N Y $

no is TSL Without Tiers

*Y

$ N N N N $ $ N N Y N $

TSL Quantifier Languages for *every* and *no*

every is TSL Without Tiers

*N

$ Y Y Y Y $ $ Y Y N Y $

no is TSL Without Tiers

*Y

$ N N N N $ $ N N Y N $

TSL Quantifier Languages for *every* and *no*

every is TSL Without Tiers

***N**

$ **Y Y Y Y** $ $ **Y Y N Y** $

no is TSL Without Tiers

***Y**

$ **N N N N** $ $ **N N Y N** $

TSL Quantifier Languages for *every* and *no*

every is TSL Without Tiers

***N**

$ **Y Y Y Y** $ $ **Y Y N Y** $

no is TSL Without Tiers

***Y**

$ **N N N N** $ $ **N N Y N** $

TSL Quantifier Languages for *every* and *no*

every is TSL Without Tiers

*N

$ Y Y Y Y $ $ Y Y N Y $

no is TSL Without Tiers

*Y

$ N N N N $ $ N N Y N $

Most Quantifier Languages Require a Tier

some is TSL With Tier {Y}

*$$

$ N N Y N $ $ N N N N $

all but 3 is TSL With Tier {N}

*$$, *N, *NN, *NNNN

$ N N Y N $ $ N Y Y N $ $ N N N N $

Most Quantifier Languages Require a Tier

some is TSL With Tier {Y}

*$$

$ Y $

| | |

$ **N N Y N** $ $ **N N N N** $

all but 3 is TSL With Tier {N}

*$$, *$**N**$, *$**NN**$, ***NNNN**

$ **N N Y N** $ $ **N Y Y N** $ $ **N N N N** $

Most Quantifier Languages Require a Tier

some is TSL With Tier {Y}

*$$

$ Y $ $ $

| | | | |

$ **N N Y N** $ $ **N N N N** $

all but 3 is TSL With Tier {N}

*$$, *$**N**$, *$**NN**$, ***NNNN**

$ **N N Y N** $ $ **N Y Y N** $ $ **N N N N** $

Most Quantifier Languages Require a Tier

some is TSL With Tier {Y}

*$$

$ Y $ $ $

$ **N N Y N** $ $ **N N N N** $

all but 3 is TSL With Tier {N}

*$$, *$**N**$, *$**NN**$, ***NNNN**

$ **N N Y N** $ $ **N Y Y N** $ $ **N N N N** $

Most Quantifier Languages Require a Tier

some is TSL With Tier {Y}

*$$

$ Y $ $ $

$ N N Y N $ $ N N N N $

all but 3 is TSL With Tier {N}

*$$, *N, *NN, *NNNN

$ N N Y N $ $ N Y Y N $ $ N N N N $

Most Quantifier Languages Require a Tier

some is TSL With Tier {Y}

*$$

$ Y $
$ N N Y N $

$ $
$ N N N N $

all but 3 is TSL With Tier {N}

*$$, *N, *NN, *NNNN

$ N N Y N $ $ N Y Y N $ $ N N N N $

Most Quantifier Languages Require a Tier

some is TSL With Tier {Y}

*$$

$ Y $ $ $

$ **N N Y N** $ $ **N N N N** $

all but 3 is TSL With Tier {N}

*$$, *$**N**$, *$**NN**$, *$**NNNN**

$ **N N Y N** $ $ **N Y Y N** $ $ **N N N N** $

Most Quantifier Languages Require a Tier

some is TSL With Tier {Y}

*$$

```
$     Y  $        $         $
|     |  |        |         |
$ N N Y N $       $ N N N N $
```

all but 3 is TSL With Tier {N}

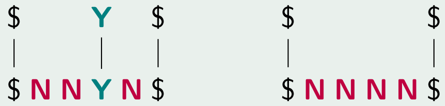
*$$, *$**N**$, *$**NN**$, ***NNNN**

```
$ N N   N $
| | |   | |
$ N N Y N $       $ N Y Y N $       $ N N N N $
```

Most Quantifier Languages Require a Tier

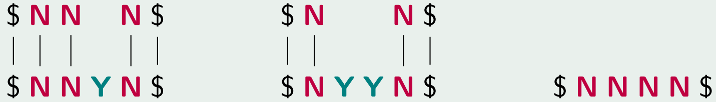
some is TSL With Tier {Y}

*$$



all but 3 is TSL With Tier {N}

*$$, *$**N**$, *$**NN**$, ***NNNN**



Most Quantifier Languages Require a Tier

some is TSL With Tier {Y}

*$$

```
$     Y  $        $         $
|     |  |        |         |
$ N N Y N $       $ N N N N $
```

all but 3 is TSL With Tier {N}

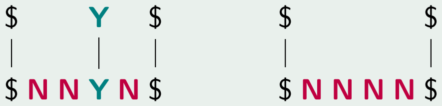
*$$, *$**N**$, *$**NN**$, ***NNNN**

```
$ N N   N $     $ N     N $     $ N N N N $
| | |   | |     | |     | |     | | | | | |
$ N N Y N $     $ N Y Y N $     $ N N N N $
```

Most Quantifier Languages Require a Tier

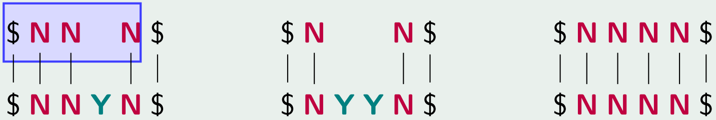
some is TSL With Tier {Y}

*$$



all but 3 is TSL With Tier {N}

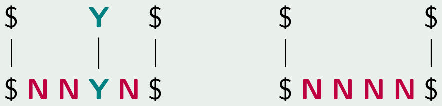
*$$, *$**N**$, *$**NN**$, ***NNNN**



Most Quantifier Languages Require a Tier

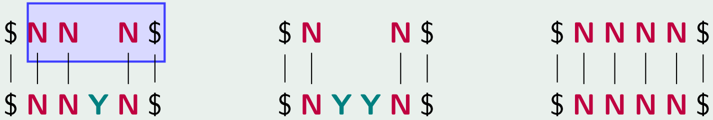
some is TSL With Tier {Y}

*$$



all but 3 is TSL With Tier {N}

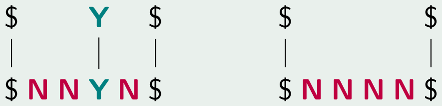
*$$, *$**N**$, *$**NN**$, ***NNNN**



Most Quantifier Languages Require a Tier

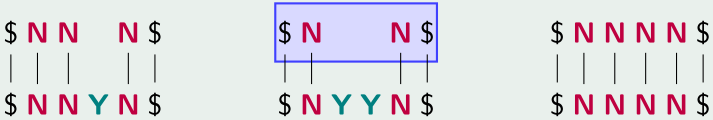
some is TSL With Tier {Y}

*$$



all but 3 is TSL With Tier {N}

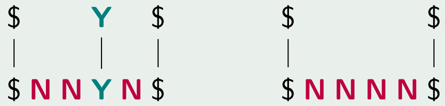
*$$, *$**N**$, *$**NN**$, ***NNNN**



Most Quantifier Languages Require a Tier

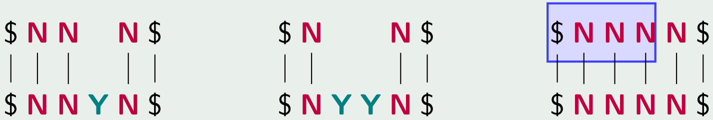
some is TSL With Tier {Y}

*$$



all but 3 is TSL With Tier {N}

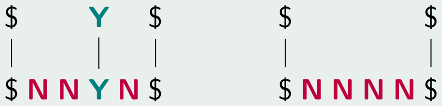
*$$, *$**N**$, *$**NN**$, ***NNNN**



Most Quantifier Languages Require a Tier

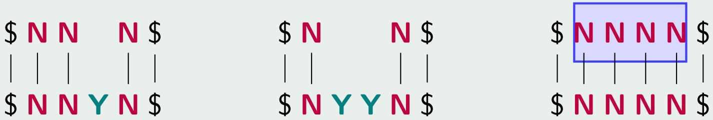
some is TSL With Tier {Y}

*$$



all but 3 is TSL With Tier {N}

*$$, *$**N**$, *$**NN**$, ***NNNN**



TSL Descriptions for Quantifier Languages

Quantifier	Constraint	n-grams	Tier
every	$\|N\| = 0$	$*N$	none
no	$\|Y\| = 0$	$*Y$	none
some	$\|Y\| \geq 1$	$*\$\$$	Y
at least n	$\|Y\| \geq n$	$*\$Y^m\$$ ($m < n$)	Y
at most n	$\|Y\| \leq n$	$*Y^{n+1}$	Y
exactly n	$\|Y\| = n$	at least + at most	Y
not all	$\|N\| \geq 1$	$*\$\$$	N
all but n	$\|N\| = n$	at least + at most	N

Typology of Quantifiers

(Paperno 2011)

Quantifier	TSL?	Tier	Mono.
every	yes	none	yes
no	yes	none	yes
some	yes	Y	yes
(at least) two	yes	Y	yes
(at most) two	yes	Y	yes
not all	yes	N	no
all but one	yes	N	no
even number	no		no
prime number	no		no
infinitely many	no		no
most	no		???

Typology of Quantifiers

(Paperno 2011)

Quantifier	TSL?	Tier	Mono.
every	yes	none	yes
no	yes	none	yes
some	yes	Y	yes
(at least) two	yes	Y	yes
(at most) two	yes	Y	yes
not all	yes	N	no
all but one	yes	N	no
even number	no		no
prime number	no		no
infinitely many	no		no
most	no		???

Typology of Quantifiers

(Paperno 2011)

Quantifier	TSL?	Tier	Mono.
every	yes	none	yes
no	yes	none	yes
some	yes	Y	yes
(at least) two	yes	Y	yes
(at most) two	yes	Y	yes
not all	yes	N	no
all but one	yes	N	no
even number	no		no
prime number	no		no
infinitely many	no		no
most	no		???

Typology of Quantifiers

(Paperno 2011)

Quantifier	TSL?	Tier	Mono.
every	yes	none	yes
no	yes	none	yes
some	yes	Y	yes
(at least) two	yes	Y	yes
(at most) two	yes	Y	yes
not all	yes	N	no
all but one	yes	N	no
even number	no		no
prime number	no		no
infinitely many	no		no
most	no		???

The Case of *most*

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

Quantifier	TSL?	Tier	Mono.
every	yes	none	yes
no	yes	none	yes
some	yes	Y	yes
(at least) two	yes	Y	yes
(at most) two	yes	Y	yes
not all	yes	N	no
all but one	yes	N	no
even number	no		no
prime number	no		no
infinitely many	no		no
most	no		no

A New Upper Bound on Typological Variation

TSL Interpretation Conjecture

If a language uses a quantifier as a monomorphemic determiner, then its quantifier language must be TSL.

TSL is Too Large

- All monomorphemic quantifiers are TSL.
- But not all TSL-definable quantifiers are monomorphemic.
- Why might that be?

Quantifier	TSL?	Tier	Mono.
every	yes	none	yes
no	yes	none	yes
some	yes	Y	yes
(at least) two	yes	Y	yes
(at most) two	yes	Y	yes
not all	yes	N	no
all but one	yes	N	no

Monotonicity

Definition (Monotonicity)

- Let **A** and **B** be two sets with orders $\leq_{\mathbf{A}}$ and $\leq_{\mathbf{B}}$, respectively.
- A function f from **A** to **B** is **monotonic** iff

$$x \leq_{\mathbf{A}} y \Rightarrow \mathsf{f}(x) \leq_{\mathbf{B}} \mathsf{f}(y)$$

- Monotonicity is similar to **No Crossing Branches** constraint.

1	2	3
A	B	C

Monotonicity

Definition (Monotonicity)

- Let **A** and **B** be two sets with orders $\leq_{\mathsf{A}}$ and $\leq_{\mathsf{B}}$, respectively.
- A function f from **A** to **B** is **monotonic** iff

$$x \leq_{\mathsf{A}} y \Rightarrow \mathsf{f}(x) \leq_{\mathsf{B}} \mathsf{f}(y)$$

- Monotonicity is similar to **No Crossing Branches** constraint.

1	2	3
|	|	|
A	B	C

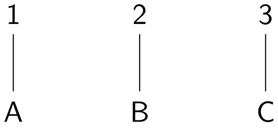
Monotonicity

Definition (Monotonicity)

- Let **A** and **B** be two sets with orders $\leq_{\mathbf{A}}$ and $\leq_{\mathbf{B}}$, respectively.
- A function f from **A** to **B** is **monotonic** iff

$$x \leq_{\mathbf{A}} y \Rightarrow \mathsf{f}(x) \leq_{\mathbf{B}} \mathsf{f}(y)$$

- Monotonicity is similar to **No Crossing Branches** constraint.



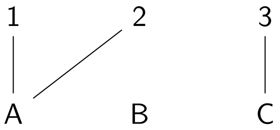
Monotonicity

Definition (Monotonicity)

- Let **A** and **B** be two sets with orders $\leq_{\mathbf{A}}$ and $\leq_{\mathbf{B}}$, respectively.
- A function **f** from **A** to **B** is **monotonic** iff

$$x \leq_{\mathbf{A}} y \Rightarrow \mathsf{f}(x) \leq_{\mathbf{B}} \mathsf{f}(y)$$

- Monotonicity is similar to **No Crossing Branches** constraint.



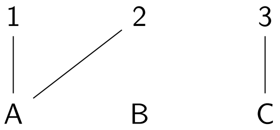
Monotonicity

Definition (Monotonicity)

- Let **A** and **B** be two sets with orders $\leq_{\mathsf{A}}$ and $\leq_{\mathsf{B}}$, respectively.
- A function f from **A** to **B** is **monotonic** iff

$$x \leq_{\mathsf{A}} y \Rightarrow \mathsf{f}(x) \leq_{\mathsf{B}} \mathsf{f}(y)$$

- Monotonicity is similar to **No Crossing Branches** constraint.



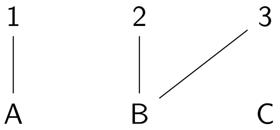
Monotonicity

Definition (Monotonicity)

- Let **A** and **B** be two sets with orders $\leq_{\mathbf{A}}$ and $\leq_{\mathbf{B}}$, respectively.
- A function f from **A** to **B** is **monotonic** iff

$$x \leq_{\mathbf{A}} y \Rightarrow \mathsf{f}(x) \leq_{\mathbf{B}} \mathsf{f}(y)$$

- Monotonicity is similar to **No Crossing Branches** constraint.



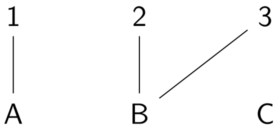
Monotonicity

Definition (Monotonicity)

- Let **A** and **B** be two sets with orders $\leq_{\mathbf{A}}$ and $\leq_{\mathbf{B}}$, respectively.
- A function **f** from **A** to **B** is **monotonic** iff

$$x \leq_{\mathbf{A}} y \Rightarrow \mathbf{f}(x) \leq_{\mathbf{B}} \mathbf{f}(y)$$

- Monotonicity is similar to **No Crossing Branches** constraint.



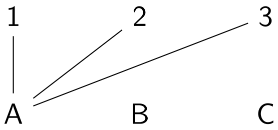
Monotonicity

Definition (Monotonicity)

- Let **A** and **B** be two sets with orders $\leq_{\mathbf{A}}$ and $\leq_{\mathbf{B}}$, respectively.
- A function f from **A** to **B** is **monotonic** iff

$$x \leq_{\mathbf{A}} y \Rightarrow \mathsf{f}(x) \leq_{\mathbf{B}} \mathsf{f}(y)$$

- Monotonicity is similar to **No Crossing Branches** constraint.



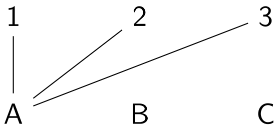
Monotonicity

Definition (Monotonicity)

- Let **A** and **B** be two sets with orders $\leq_{\mathbf{A}}$ and $\leq_{\mathbf{B}}$, respectively.
- A function f from **A** to **B** is **monotonic** iff

$$x \leq_{\mathbf{A}} y \Rightarrow \mathsf{f}(x) \leq_{\mathbf{B}} \mathsf{f}(y)$$

- Monotonicity is similar to **No Crossing Branches** constraint.



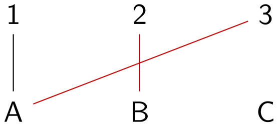
Monotonicity

Definition (Monotonicity)

- Let **A** and **B** be two sets with orders $\leq_{\mathsf{A}}$ and $\leq_{\mathsf{B}}$, respectively.
- A function f from **A** to **B** is **monotonic** iff

$$x \leq_{\mathsf{A}} y \Rightarrow \mathsf{f}(x) \leq_{\mathsf{B}} \mathsf{f}(y)$$

- Monotonicity is similar to **No Crossing Branches** constraint.



Monotonicity in Language

- **Monotonicity in phonology**
  - No Crossing Branches constraint
  - Natural classes are convex
- **Monotonicity in morphology**
  - *ABA
- **Monotonicity in syntax**
  - Subcategorization $<$ A-Move $<$ A′-Move
  - Adjunct Island Constraint & Coordinate Structure Constraint
- **Monotonicity in semantics**
  - Everywhere. . .

Montonicity in Tier Projection

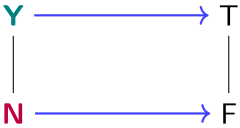
- Suppose, then, that monotonicity is a desirable trait.
- How does monotonicity relate to tier projection?

Project:

- Monotonicity forbids projecting only **N**.

Montonicity in Tier Projection

- Suppose, then, that monotonicity is a desirable trait.
- How does monotonicity relate to tier projection?

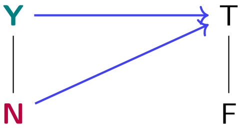


Project: Y

- Monotonicity forbids projecting only N.

Montonicity in Tier Projection

- Suppose, then, that monotonicity is a desirable trait.
- How does monotonicity relate to tier projection?

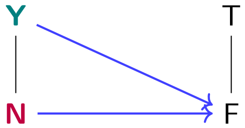


Project: Yand N

- Monotonicity forbids projecting only N.

Montonicity in Tier Projection

- Suppose, then, that monotonicity is a desirable trait.
- How does monotonicity relate to tier projection?

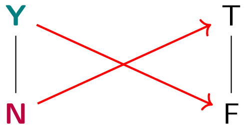


Project: nothing

- Monotonicity forbids projecting only **N**.

Montonicity in Tier Projection

- Suppose, then, that monotonicity is a desirable trait.
- How does monotonicity relate to tier projection?

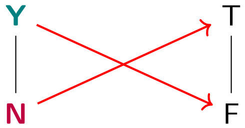


Project: forbidden

- Monotonicity forbids projecting only **N**.

Montonicity in Tier Projection

- Suppose, then, that monotonicity is a desirable trait.
- How does monotonicity relate to tier projection?



Project: forbidden

- Monotonicity forbids projecting only **N**.

Adding Tiers for *every* and *no*

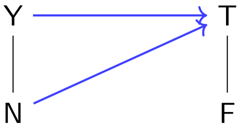
- *every* and *no* are the only quantifiers without tier
- **But:** no tier = tier containing everything

Example

$ N N Y Y N N $

$ N N Y Y N N $
| | | | | | | |
$ N N Y Y N N $

- So *every* and *no* can be viewed as using the tier {Y,N}.
- This satisfies monotonicity.



Remaining Issues & Extensions

- TSL also allows for some unnatural quantifiers;
  ruling them out requires some stipulations.
- What about fuzzy quantifiers?
  many, few, ...
- TSL makes cognitive complexity predictions;
  we're working on experiments.
- Where else in semantics does TSL matter?
  - adverbial quantifiers
  - temporal semantics
  - modals
- But those are just small pieces of a **much larger puzzle**...

The Bigger Goal

- Computational approaches are abstract and content-neutral.
- This isn't a problem but a **virtue**.
- Abstraction makes it possible to identify parallels between very different domains.

A Program of Subregular Unification

- To what extent can very different properties of language be reduced to the same computational property?
- What are the implications for
  - typological variation,
  - learnability,
  - cognition at large?

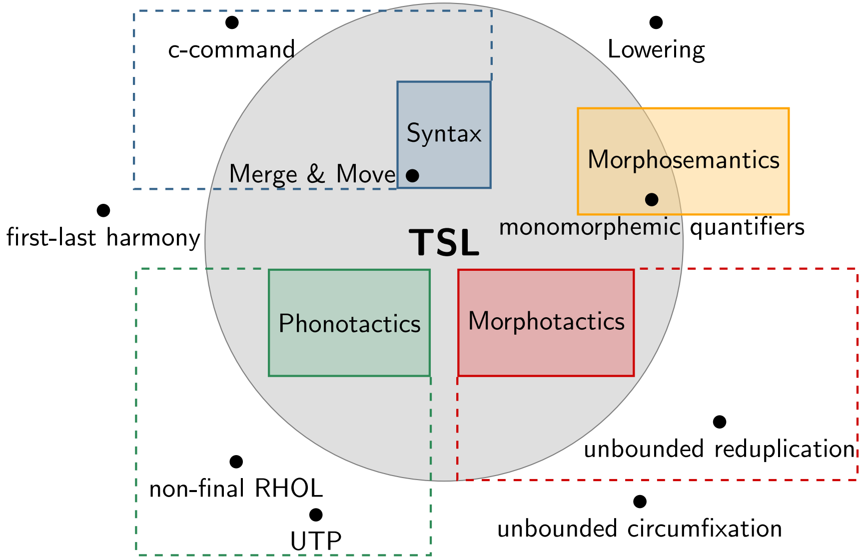
The Bigger Goal

- Computational approaches are abstract and content-neutral.
- This isn't a problem but a **virtue**.
- Abstraction makes it possible to identify parallels between very different domains.

A Program of Subregular Unification

- To what extent can very different properties of language be reduced to the same computational property?
- What are the implications for
  - typological variation,
  - learnability,
  - cognition at large?

Place of Morphosemantics



Conclusion

- Among determiners, all monomorphemic quantifiers have quantifier languages that are TSL.
- The opposite does not hold, additional restrictions on TSL are needed.
- Why does it matter? Because TSL is everywhere in language.
- Ultimate goal:
  computational explanation of typological variation

References I

van Benthem, Johan. 1986. Semantic automata. In *Essays in logical semantics*, 151–176. Dordrecht: Springer.

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

Fernando, Tim. 2011. Regular relations for temporal propositions. *Natural Language Engineering* 11:163–184.

Goldsmith, John. 1976. *Autosegmental phonology*. Doctoral Dissertation, MIT.

Graf, Thomas. 2017. The power of locality domains in phonology. *Phonology* In press.

Hackl, Martin. 2009. On the grammar and processing of proportional quantifiers: Most versus more than half. *Natural Language Semantics* 17:63–98.

Heinz, Jeffrey. 2009. On the role of locality in learning stress patterns. *Phonology* 26:303–351. URL `https://doi.org/10.1017/S0952675709990145`.

Heinz, Jeffrey. 2010. Learning long-distance phonotactics. *Linguistic Inquiry* 41:623–661. URL `http://dx.doi.org/10.1162/LING_a_00015`.

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`.

References II

Jardine, Adam. 2016. Computationally, tone is different. *Phonology* URL http://udel.edu/~ajardine/files/jardinemscomputationallytoneisdifferent.pdf, to appear.

Jardine, Adam, and Kevin McMullin. 2017. Efficient learning of tier-based strictly k-local languages. In *Proceedings of Language and Automata Theory and Applications*, Lecture Notes in Computer Science, 64–76. Springer.

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

Paperno, Denis. 2011. Learnable classes of natural language quantifiers: Two perspectives. URL http://paperno.bol.ucla.edu/q_learning.pdf, ms., UCLA.