# A Tier-Based Model of Syntactic Agreement

Kenneth Hanson

Stony Brook University

CLS 60

April 26–28, 2024

# Some (paradoxical) properties of agreement

| Usually... | but... |
|---|---|
| Applies over a distance | Subject to blockers |
| Blockers are predictable | Vary across dependencies/languages |
| Targets the closest visible DP | Which DPs are visible varies |
| Probe c-commands goal | Sometimes reversed |
| One probe $\leftrightarrow$ one goal | Sometimes many-to-one |

# Overview

**Main claim:** Agreement patterns are **tier-based strictly local (TSL)**, mirroring findings on movement (Graf 2022b) and case (Hanson 2023b).

# Overview

**Main claim:** Agreement patterns are **tier-based strictly local (TSL)**, mirroring findings on movement (Graf 2022b) and case (Hanson 2023b).

**Why this matters:**

- Limits structural configurations

# Overview

**Main claim:** Agreement patterns are **tier-based strictly local (TSL)**, mirroring findings on movement (Graf 2022b) and case (Hanson 2023b).

**Why this matters:**

- Limits structural configurations
- Defines parameters for variation

# Overview

**Main claim:** Agreement patterns are **tier-based strictly local (TSL)**, mirroring findings on movement (Graf 2022b) and case (Hanson 2023b).

**Why this matters:**

- Limits structural configurations
- Defines parameters for variation
- Provides a unified model of locality restrictions

# Overview

**Main claim:** Agreement patterns are **tier-based strictly local (TSL)**, mirroring findings on movement (Graf 2022b) and case (Hanson 2023b).

**Why this matters:**

- Limits structural configurations
- Defines parameters for variation
- Provides a unified model of locality restrictions
- Shows parallels within/across domains

# Overview

**Main claim:** Agreement patterns are **tier-based strictly local (TSL)**, mirroring findings on movement (Graf 2022b) and case (Hanson 2023b).

**Why this matters:**

- Limits structural configurations
- Defines parameters for variation
- Provides a unified model of locality restrictions
- Shows parallels within/across domains
- Derives typology from issues of efficient computation

# Roadmap

1. What is a TSL pattern?
2. A TSL model of agreement
3. Consequences for locality
4. Typological variation
5. Parallels with phonology
6. Strengths and limitations of the model

# What is a TSL pattern?

1. Ignore the irrelevant items and treat the rest as if adjacent
2. All constraints must be stated within a fixed-size moving window

# What is a TSL pattern?

1. Ignore the irrelevant items and treat the rest as if adjacent
2. All constraints must be stated within a fixed-size moving window

**Example: Vowel harmony**

i/u/o obey front-back harmony, *e* is transparent/neutral, *a* is opaque

# What is a TSL pattern?

1. Ignore the irrelevant items and treat the rest as if adjacent
2. All constraints must be stated within a fixed-size moving window

**Example: Vowel harmony**

i/u/o obey front-back harmony, *e* is transparent/neutral, *a* is opaque

Tier elements: {i, u, o, a} Constraints: {*iu, *ui, *io, *oi}

# What is a TSL pattern?

1. Ignore the irrelevant items and treat the rest as if adjacent
2. All constraints must be stated within a fixed-size moving window

**Example: Vowel harmony**

i/u/o obey front-back harmony, *e* is transparent/neutral, *a* is opaque

Tier elements: {i, u, o, a}  Constraints: {*iu, *ui, *io, *oi}

k u b u l o

# What is a TSL pattern?

1. Ignore the irrelevant items and treat the rest as if adjacent
2. All constraints must be stated within a fixed-size moving window

**Example: Vowel harmony**

i/u/o obey front-back harmony, *e* is transparent/neutral, *a* is opaque

Tier elements: {i, u, o, a} Constraints: {*iu, *ui, *io, *oi}

k u b u l o

# What is a TSL pattern?

1. Ignore the irrelevant items and treat the rest as if adjacent
2. All constraints must be stated within a fixed-size moving window

**Example: Vowel harmony**

i/u/o obey front-back harmony, *e* is transparent/neutral, *a* is opaque

Tier elements: {i, u, o, a} Constraints: {*iu, *ui, *io, *oi}

k u b u l o

# What is a TSL pattern?

1. Ignore the irrelevant items and treat the rest as if adjacent
2. All constraints must be stated within a fixed-size moving window

**Example: Vowel harmony**

i/u/o obey front-back harmony, *e* is transparent/neutral, *a* is opaque

Tier elements: {i, u, o, a}    Constraints: {*iu, *ui, *io, *oi}

k u b $\boxed{u}$ l $\boxed{o}$

# What is a TSL pattern?

1. Ignore the irrelevant items and treat the rest as if adjacent
2. All constraints must be stated within a fixed-size moving window

**Example: Vowel harmony**

i/u/o obey front-back harmony, *e* is transparent/neutral, *a* is opaque

Tier elements: {i, u, o, a}     Constraints: {*iu, *ui, *io, *oi}

k u b u l o ✓

# What is a TSL pattern?

1. Ignore the irrelevant items and treat the rest as if adjacent
2. All constraints must be stated within a fixed-size moving window

**Example: Vowel harmony**

i/u/o obey front-back harmony, *e* is transparent/neutral, *a* is opaque

Tier elements: {i, u, o, a}    Constraints: {*iu, *ui, *io, *oi}

k u b u l o ✓    k i b i l o

# What is a TSL pattern?

1. Ignore the irrelevant items and treat the rest as if adjacent
2. All constraints must be stated within a fixed-size moving window

**Example: Vowel harmony**

i/u/o obey front-back harmony, *e* is transparent/neutral, *a* is opaque

Tier elements: {i, u, o, a} Constraints: {*iu, *ui, *io, *oi}

k u b u l o ✓ k i b i l o

# What is a TSL pattern?

1. Ignore the irrelevant items and treat the rest as if adjacent
2. All constraints must be stated within a fixed-size moving window

**Example: Vowel harmony**

i/u/o obey front-back harmony, *e* is transparent/neutral, *a* is opaque

Tier elements: {i, u, o, a} Constraints: {*iu, *ui, *io, *oi}

k u b u l o ✓ k $\boxed{\text{i}}$ b $\boxed{\text{i}}$ l o

# What is a TSL pattern?

1. Ignore the irrelevant items and treat the rest as if adjacent
2. All constraints must be stated within a fixed-size moving window

**Example: Vowel harmony**

i/u/o obey front-back harmony, *e* is transparent/neutral, *a* is opaque

Tier elements: {i, u, o, a} Constraints: {*iu, *ui, *io, *oi}

k u b u l o ✓ k i b $\boxed{\text{i}}$ l $\boxed{\text{o}}$

# What is a TSL pattern?

1. Ignore the irrelevant items and treat the rest as if adjacent
2. All constraints must be stated within a fixed-size moving window

**Example: Vowel harmony**

i/u/o obey front-back harmony, *e* is transparent/neutral, *a* is opaque

Tier elements: {i, u, o, a} Constraints: {*iu, *ui, *io, *oi}

k u b u l o ✓ k i b i l o ✗

# What is a TSL pattern?

1. Ignore the irrelevant items and treat the rest as if adjacent
2. All constraints must be stated within a fixed-size moving window

**Example: Vowel harmony**

i/u/o obey front-back harmony, *e* is transparent/neutral, *a* is opaque

Tier elements: {i, u, o, a} Constraints: {*iu, *ui, *io, *oi}

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| k | u | b | u | l | o | ✓ | k | i | b | i | l | o | ✗ |
| k | u | b | e | l | o | | | | | | | | |

# What is a TSL pattern?

1. Ignore the irrelevant items and treat the rest as if adjacent
2. All constraints must be stated within a fixed-size moving window

**Example: Vowel harmony**

i/u/o obey front-back harmony, *e* is transparent/neutral, *a* is opaque

Tier elements: {i, u, o, a} Constraints: {*iu, *ui, *io, *oi}

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| k | u | b | u | l | o | ✓ | | k | i | b | i | l | o | ✗ |
| k | u | b | e | l | o | | | | | | | | | |

# What is a TSL pattern?

1. Ignore the irrelevant items and treat the rest as if adjacent
2. All constraints must be stated within a fixed-size moving window

**Example: Vowel harmony**

i/u/o obey front-back harmony, *e* is transparent/neutral, *a* is opaque

Tier elements: {i, u, o, a} Constraints: {*iu, *ui, *io, *oi}

k u b u l o ✓ k i b i l o ✗

k u b e l o

# What is a TSL pattern?

1. Ignore the irrelevant items and treat the rest as if adjacent
2. All constraints must be stated within a fixed-size moving window

**Example: Vowel harmony**

i/u/o obey front-back harmony, *e* is transparent/neutral, *a* is opaque

Tier elements: {i, u, o, a} Constraints: {*iu, *ui, *io, *oi}

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| k | u | b | u | l | o | ✓ | k | i | b | i | l | o | ✗ |
| k | u | b | e | l | o | ✓ | | | | | | | |

# What is a TSL pattern?

1. Ignore the irrelevant items and treat the rest as if adjacent
2. All constraints must be stated within a fixed-size moving window

**Example: Vowel harmony**

i/u/o obey front-back harmony, *e* is transparent/neutral, *a* is opaque

Tier elements: {i, u, o, a} Constraints: {*iu, *ui, *io, *oi}

k u b u l o ✓ k i b i l o ✗

k u b e l o ✓ k i b e l o

# What is a TSL pattern?

1. Ignore the irrelevant items and treat the rest as if adjacent
2. All constraints must be stated within a fixed-size moving window

**Example: Vowel harmony**

i/u/o obey front-back harmony, *e* is transparent/neutral, *a* is opaque

Tier elements: {i, u, o, a} Constraints: {*iu, *ui, *io, *oi}

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| k | u | b | u | l | o | ✓ | | k | i | b | i | l | o | ✗ |
| k | u | b | e | l | o | ✓ | | k | i | b | e | l | o | |

# What is a TSL pattern?

1. Ignore the irrelevant items and treat the rest as if adjacent
2. All constraints must be stated within a fixed-size moving window

**Example: Vowel harmony**

i/u/o obey front-back harmony, *e* is transparent/neutral, *a* is opaque

Tier elements: {i, u, o, a}    Constraints: {*iu, *ui, *io, *oi}

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| k | u | b | u | l | o | ✓ | k | i | b | i | l | o | ✗ |
| k | u | b | e | l | o | ✓ | k | i | b | e | l | o | |

# What is a TSL pattern?

1. Ignore the irrelevant items and treat the rest as if adjacent
2. All constraints must be stated within a fixed-size moving window

**Example: Vowel harmony**

i/u/o obey front-back harmony, *e* is transparent/neutral, *a* is opaque

Tier elements: {i, u, o, a} Constraints: {*iu, *ui, *io, *oi}

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| k | u | b | u | l | o | ✓ | | k | i | b | i | l | o | ✗ |
| k | u | b | e | l | o | ✓ | | k | i | b | e | l | o | ✗ |

# What is a TSL pattern?

1. Ignore the irrelevant items and treat the rest as if adjacent
2. All constraints must be stated within a fixed-size moving window

**Example: Vowel harmony**

i/u/o obey front-back harmony, *e* is transparent/neutral, *a* is opaque

Tier elements: {i, u, o, a} Constraints: {*iu, *ui, *io, *oi}

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| k | u | b | u | l | o | ✓ | | k | i | b | i | l | o | ✗ |
| k | u | b | e | l | o | ✓ | | k | i | b | e | l | o | ✗ |
| k | u | b | a | l | o | | | | | | | | | |

# What is a TSL pattern?

1. Ignore the irrelevant items and treat the rest as if adjacent
2. All constraints must be stated within a fixed-size moving window

**Example: Vowel harmony**

i/u/o obey front-back harmony, *e* is transparent/neutral, *a* is opaque

Tier elements: {i, u, o, a}    Constraints: {*iu, *ui, *io, *oi}

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| k | u | b | u | l | o | ✓ | | k | i | b | i | l | o | ✗ |
| k | u | b | e | l | o | ✓ | | k | i | b | e | l | o | ✗ |
| k | u | b | a | l | o | | | | | | | | | |

# What is a TSL pattern?

1. Ignore the irrelevant items and treat the rest as if adjacent
2. All constraints must be stated within a fixed-size moving window

**Example: Vowel harmony**

i/u/o obey front-back harmony, *e* is transparent/neutral, *a* is opaque

Tier elements: {i, u, o, a} Constraints: {*iu, *ui, *io, *oi}

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| k | u | b | u | l | o | ✓ | k | i | b | i | l | o | ✗ |
| k | u | b | e | l | o | ✓ | k | i | b | e | l | o | ✗ |
| k | $\boxed{u}$ | b | $\boxed{a}$ | l | o | | | | | | | | |

# What is a TSL pattern?

1. Ignore the irrelevant items and treat the rest as if adjacent
2. All constraints must be stated within a fixed-size moving window

**Example: Vowel harmony**

i/u/o obey front-back harmony, *e* is transparent/neutral, *a* is opaque

Tier elements: {i, u, o, a}  Constraints: {*iu, *ui, *io, *oi}

k u b u l o ✓  k i b i l o ✗

k u b e l o ✓  k i b e l o ✗

k u b $\boxed{\text{a}}$ l $\boxed{\text{o}}$

# What is a TSL pattern?

1. Ignore the irrelevant items and treat the rest as if adjacent
2. All constraints must be stated within a fixed-size moving window

**Example: Vowel harmony**

i/u/o obey front-back harmony, *e* is transparent/neutral, *a* is opaque

Tier elements: {i, u, o, a} Constraints: {*iu, *ui, *io, *oi}

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| k | u | b | u | l | o | ✓ | | k | i | b | i | l | o | ✗ |
| k | u | b | e | l | o | ✓ | | k | i | b | e | l | o | ✗ |
| k | u | b | a | l | o | ✓ | | | | | | | | |

# What is a TSL pattern?

1. Ignore the irrelevant items and treat the rest as if adjacent
2. All constraints must be stated within a fixed-size moving window

**Example: Vowel harmony**

i/u/o obey front-back harmony, *e* is transparent/neutral, *a* is opaque

Tier elements: {i, u, o, a} Constraints: {*iu, *ui, *io, *oi}

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| k | u | b | u | l | o | ✓ | | k | i | b | i | l | o | ✗ |
| k | u | b | e | l | o | ✓ | | k | i | b | e | l | o | ✗ |
| k | u | b | a | l | o | ✓ | | k | i | b | a | l | o | |

# What is a TSL pattern?

1. Ignore the irrelevant items and treat the rest as if adjacent
2. All constraints must be stated within a fixed-size moving window

**Example: Vowel harmony**

i/u/o obey front-back harmony, *e* is transparent/neutral, *a* is opaque

Tier elements: {i, u, o, a} Constraints: {*iu, *ui, *io, *oi}

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| k | u | b | u | l | o | ✓ | k | i | b | i | l | o | ✗ |
| k | u | b | e | l | o | ✓ | k | i | b | e | l | o | ✗ |
| k | u | b | a | l | o | ✓ | k | i | b | a | l | o | |

# What is a TSL pattern?

1. Ignore the irrelevant items and treat the rest as if adjacent
2. All constraints must be stated within a fixed-size moving window

**Example: Vowel harmony**

i/u/o obey front-back harmony, *e* is transparent/neutral, *a* is opaque

Tier elements: {i, u, o, a} Constraints: {*iu, *ui, *io, *oi}

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| k | u | b | u | l | o | ✓ | k | i | b | i | l | o | ✗ |
| k | u | b | e | l | o | ✓ | k | i | b | e | l | o | ✗ |
| k | u | b | a | l | o | ✓ | k | i | b | a | l | o | |

# What is a TSL pattern?

1. Ignore the irrelevant items and treat the rest as if adjacent
2. All constraints must be stated within a fixed-size moving window

**Example: Vowel harmony**

i/u/o obey front-back harmony, *e* is transparent/neutral, *a* is opaque

Tier elements: {i, u, o, a} Constraints: {*iu, *ui, *io, *oi}

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| k | u | b | u | l | o | ✓ | | k | i | b | i | l | o | ✗ |
| k | u | b | e | l | o | ✓ | | k | i | b | e | l | o | ✗ |
| k | u | b | a | l | o | ✓ | | k | i | b | $\boxed{a}$ | l | $\boxed{o}$ | |

# What is a TSL pattern?

1. Ignore the irrelevant items and treat the rest as if adjacent
2. All constraints must be stated within a fixed-size moving window

**Example: Vowel harmony**

i/u/o obey front-back harmony, *e* is transparent/neutral, *a* is opaque

Tier elements: {i, u, o, a} Constraints: {*iu, *ui, *io, *oi}

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| k | u | b | u | l | o | ✓ | | k | i | b | i | l | o | ✗ |
| k | u | b | e | l | o | ✓ | | k | i | b | e | l | o | ✗ |
| k | u | b | a | l | o | ✓ | | k | i | b | a | l | o | ✓ |

# More about TSL

- Originally defined to model phonological patterns (Heinz et al. 2011)
- Argued to be relevant in syntax as well (Graf 2022a)
- Inspired by but distinct from autosegmental phonology (Goldsmith 1976)
- Special relational structure (tier successor) with very weak constraint logic (banned substrings) (Lambert et al. 2021)
- By hypothesis, we only need a window of size two (McMullin 2016)

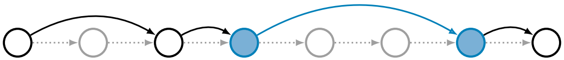


Figure 1: TSL string model with constraint window of size two

See Appendix 1 for another example and a formal definition.

# A TSL Model of Agreement

# Setup

**Assumptions:**

- Bare phrase structure, feature-driven selection, movement, ...
- Agreement between elements with initially unvalued features (probes) and elements which provide those values (goals)

# Setup

**Assumptions:**

- Bare phrase structure, feature-driven selection, movement, ...
- Agreement between elements with initially unvalued features (probes) and elements which provide those values (goals)

**Question:** What are the possible arrangements of probes and goals for agreement?

# Setup

**Assumptions:**

- Bare phrase structure, feature-driven selection, movement, ...
- Agreement between elements with initially unvalued features (probes) and elements which provide those values (goals)

**Question:** What are the possible arrangements of probes and goals for agreement?

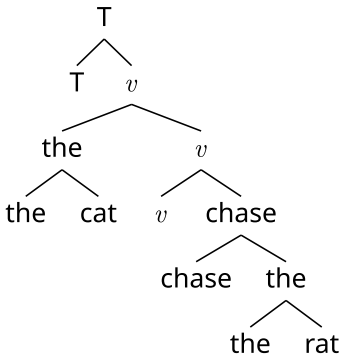
**Answer:** They are TSL constraints on the search path of the probe.

# The search path

The search path follows the **derivational command (d-command)** relation (Graf and Shafiei 2019).

- Head < Spec < Comp
- d-command order $\approx$ height of XP
  $\approx$ order of last merge
  $\approx$ reverse order of selection
- Projections of a head are not distinguished.
- At each branching point, follow the complement spine (Graf and De Santo 2019).

ex. ‘The cat chases the rats.’



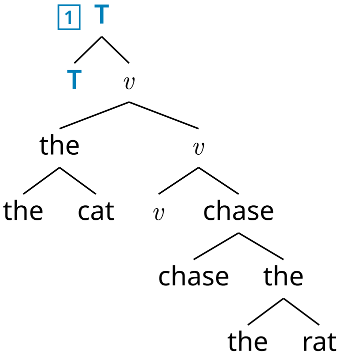

See Appendix 2 for how this works using derivation trees.

# The search path

The search path follows the **derivational command (d-command)** relation (Graf and Shafiei 2019).

- Head < Spec < Comp
- d-command order ≈ height of XP
  ≈ order of last merge
  ≈ reverse order of selection
- Projections of a head are not distinguished.
- At each branching point, follow the complement spine (Graf and De Santo 2019).

ex. ‘The cat chases the rats.’



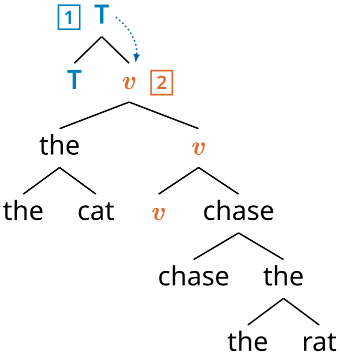

See Appendix 2 for how this works using derivation trees.

# The search path

The search path follows the **derivational command (d-command)** relation (Graf and Shafiei 2019).

- Head < Spec < Comp
- d-command order ≈ height of XP
  ≈ order of last merge
  ≈ reverse order of selection
- Projections of a head are not distinguished.
- At each branching point, follow the complement spine (Graf and De Santo 2019).

ex. 'The cat chases the rats.'



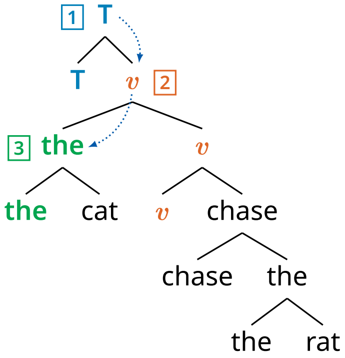

See Appendix 2 for how this works using derivation trees.

# The search path

The search path follows the **derivational command (d-command)** relation (Graf and Shafiei 2019).

- Head < Spec < Comp
- d-command order ≈ height of XP
  ≈ order of last merge
  ≈ reverse order of selection
- Projections of a head are not distinguished.
- At each branching point, follow the complement spine (Graf and De Santo 2019).

ex. ‘The cat chases the rats.’



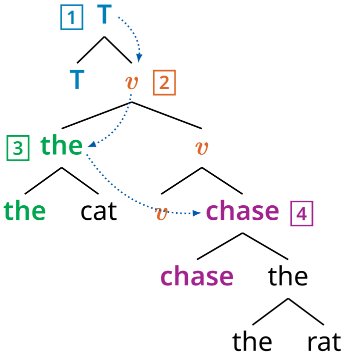

See Appendix 2 for how this works using derivation trees.

# The search path

The search path follows the **derivational command (d-command)** relation (Graf and Shafiei 2019).

- Head < Spec < Comp
- d-command order ≈ height of XP
  ≈ order of last merge
  ≈ reverse order of selection
- Projections of a head are not distinguished.
- At each branching point, follow the complement spine (Graf and De Santo 2019).

ex. ‘The cat chases the rats.’



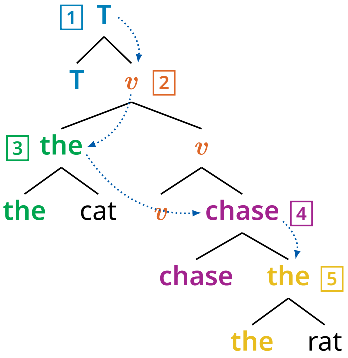

See Appendix 2 for how this works using derivation trees.

# The search path

The search path follows the **derivational command (d-command)** relation (Graf and Shafiei 2019).

- Head < Spec < Comp
- d-command order ≈ height of XP
  ≈ order of last merge
  ≈ reverse order of selection
- Projections of a head are not distinguished.
- At each branching point, follow the complement spine (Graf and De Santo 2019).

ex. 'The cat chases the rats.'



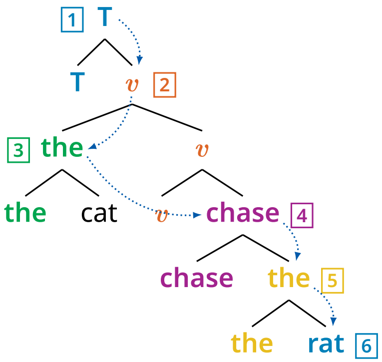

See Appendix 2 for how this works using derivation trees.

# The search path

The search path follows the **derivational command (d-command)** relation (Graf and Shafiei 2019).

- Head < Spec < Comp
- d-command order ≈ height of XP
  ≈ order of last merge
  ≈ reverse order of selection
- Projections of a head are not distinguished.
- At each branching point, follow the complement spine (Graf and De Santo 2019).

ex. 'The cat chases the rats.'




See Appendix 2 for how this works using derivation trees.

# The TSL analysis

**General principle:** a probe must be immediately followed by its goal on a tier projected from the search path (and vice versa).

# The TSL analysis

**General principle:** a probe must be immediately followed by its goal on a tier projected from the search path (and vice versa).

**Notation:** $p\phi$ = probe $g\phi$ = actual goal $\phi$ = other potential goal

# The TSL analysis

**General principle:** a probe must be immediately followed by its goal on a tier projected from the search path (and vice versa).

**Notation:** $p\phi$ = probe    $g\phi$ = actual goal    $\phi$ = other potential goal

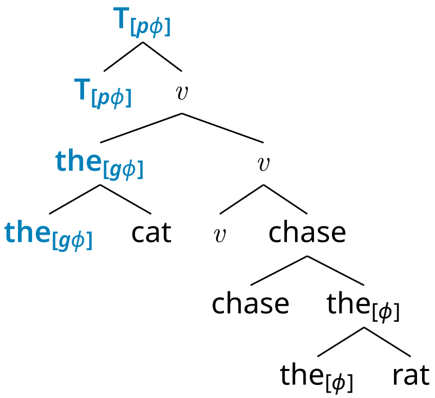
**Example: (canonical) subject-verb agreement**

| | |
|---|---|
| Tier elements: | All agreeing elements (T/D) and blockers (C) |
| Constraints: | $*\mathrm{T}_{[p\phi]} \cdot \mathrm{D}_{[\phi]}$, $*\mathrm{T}_{[p\phi]} \cdot \mathrm{C}$, $*\mathrm{D} \cdot \mathrm{D}_{[g\phi]}$, $*\mathrm{D}_{[g\phi]} \cdot \mathrm{D}_{[g\phi]}$, $\ldots$ |

# The TSL analysis – example

**General principle:** a probe must be immediately followed by its goal on a tier projected from the search path (and vice versa).

ex. The cat **chases** the rats.

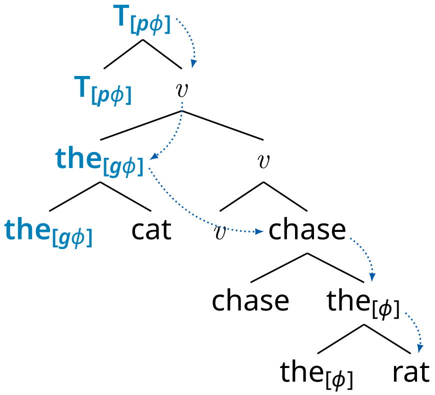



For simplicity, we substitute most items with their category labels.

# The TSL analysis – example

**General principle:** a probe must be immediately followed by its goal on a tier projected from the search path (and vice versa).

ex. The cat **chases** the rats.



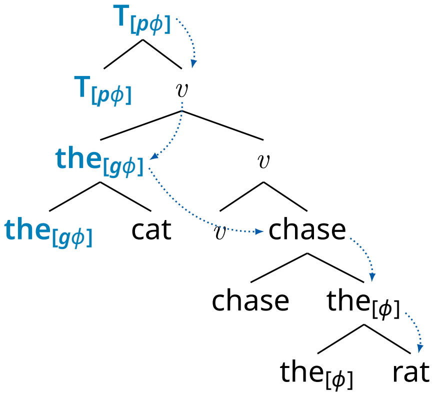

For simplicity, we substitute most items with their category labels.

# The TSL analysis – example

**General principle:** a probe must be immediately followed by its goal on a tier projected from the search path (and vice versa).

ex. The cat **chases** the rats.

Path: $\mathbf{T}_{[p\phi]} \cdot v \cdot \mathbf{D}_{[g\phi]} \cdot \mathrm{V} \cdot \mathrm{D}_{[\phi]} \cdot \mathrm{N}$



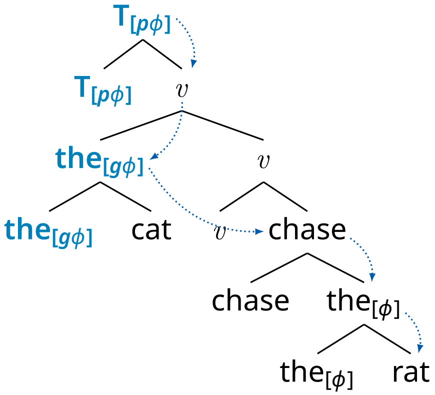

For simplicity, we substitute most items with their category labels.

# The TSL analysis – example

**General principle:** a probe must be immediately followed by its goal on a tier projected from the search path (and vice versa).

ex. The cat **chases** the rats.

Path: $\mathbf{T_{[p\phi]}} \cdot v \cdot \mathbf{D_{[g\phi]}} \cdot \mathrm{V} \cdot \mathrm{D_{[\phi]}} \cdot \mathrm{N}$




For simplicity, we substitute most items with their category labels.

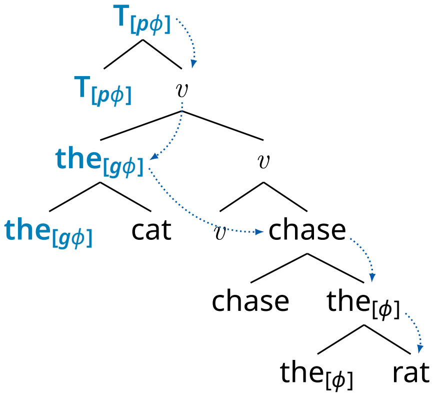
# The TSL analysis – example

**General principle:** a probe must be immediately followed by its goal on a tier projected from the search path (and vice versa).

ex. The cat **chases** the rats.

Path: $\mathbf{T_{[p\phi]}} \cdot v \cdot \mathbf{D_{[g\phi]}} \cdot \mathrm{V} \cdot \mathrm{D_{[\phi]}} \cdot \mathrm{N}$

Tier: $\mathbf{T_{[p\phi]}} \cdot \mathbf{D_{[g\phi]}} \cdot \mathrm{D_{[\phi]}}$




For simplicity, we substitute most items with their category labels.

# The TSL analysis – example

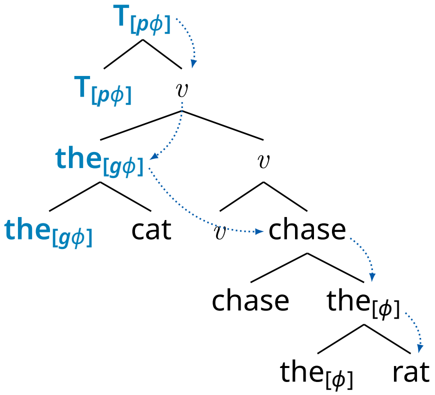
**General principle:** a probe must be immediately followed by its goal on a tier projected from the search path (and vice versa).

ex. The cat **chases** the rats.

Path: $\mathbf{T_{[p\phi]}} \cdot v \cdot \mathbf{D_{[g\phi]}} \cdot \mathrm{V} \cdot \mathrm{D_{[\phi]}} \cdot \mathrm{N}$

Tier: $\mathbf{T_{[p\phi]}} \cdot \mathbf{D_{[g\phi]}} \cdot \mathrm{D_{[\phi]}}$

Violations: n/a




For simplicity, we substitute most items with their category labels.

# Consequences for locality

# Consequences for locality

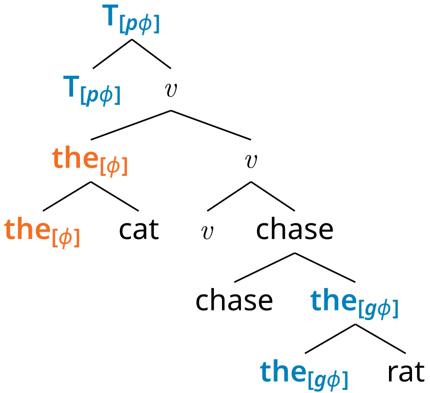
- **Minimality:** if another potential goal intervenes on the tier, agreement is blocked.
- **Invisibility:** if a DP is omitted from the tier, long-distance agreement is possible.
  - e.g. agreement across *there*, case-sensitive agreement
- **Blocking:** if a non-agreeing element intervenes on the tier, agreement is blocked.
  - e.g. probe horizons (Keine 2019), defective intervention

# Minimality

If another potential goal intervenes on the tier, agreement is blocked.

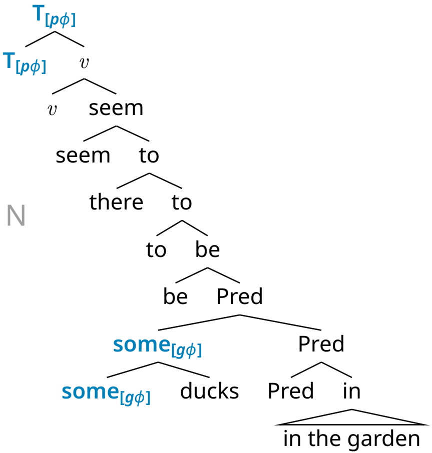
ex. * The cat **chase** <u>the rats</u>.

| | |
|---|---|
| Path: | $\mathbf{T_{[p\phi]}} \cdot v \cdot \mathbf{D_{[\phi]}} \cdot \mathrm{V} \cdot \mathbf{D_{[g\phi]}} \cdot \mathrm{N}$ |
| Tier: | $\mathbf{T_{[p\phi]}} \cdot \mathbf{D_{[\phi]}} \cdot \mathbf{D_{[g\phi]}}$ |
| Violations: | $*\mathrm{T_{[p\phi]}} \cdot \mathrm{D_{[\phi]}}$, $*\mathrm{D_{[\phi]}} \cdot \mathrm{D_{[g\phi]}}$ |



# Invisibility

If a DP is omitted from the tier, long-distance agreement is possible.




ex. There **seem** to be <u>some ducks</u> in the garden.

| | |
|---|---|
| Path: | $\mathbf{T}_{[p\phi]} \cdot v \cdot \mathrm{V} \cdot \mathrm{T} \cdot \mathrm{there} \cdot v \cdot \mathbf{D}_{[g\phi]} \cdot \mathrm{P} \cdot \mathrm{D}_{[\phi]} \cdot \mathrm{N}$ |
| Tier: | $\mathbf{T}_{[p\phi]} \cdot \mathbf{D}_{[g\phi]} \cdot \mathrm{D}_{[\phi]}$ |
| Violations: | n/a |

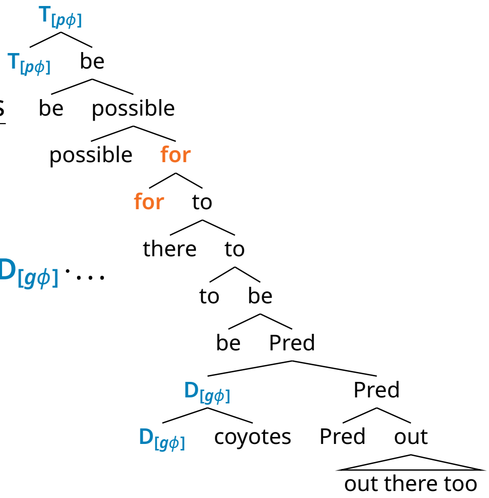
We can handle optional default agreement in several ways. Ask me if you are interested.

# Blocking

If a non-agreeing element is projected on the tier, agreement is blocked.

ex. * It **are** possible for there to be coyotes out there too.

cf. It **is** possible…




Path: $\mathbf{T}_{[p\phi]} \cdot v \cdot \mathrm{V} \cdot \mathbf{C} \cdot \mathrm{T} \cdot \text{there} \cdot v \cdot \text{Pred} \cdot \mathbf{D}_{[g\phi]} \cdot \ldots$

Tier: $\mathbf{T}_{[p\phi]} \cdot \mathbf{C} \cdot \mathbf{D}_{[g\phi]}$

Violations: $*\mathrm{T}_{[p\phi]} \cdot \mathrm{C}$, $*\mathrm{C} \cdot \mathrm{D}_{[g\phi]}$

Assume for the sake of demonstration that expletive "it" is inserted late and does not agree.

# Locality – summary

Locality phenomena derive from TSL with a window of size two, a.k.a. **TSL-2**.

- Minimality: closer potential goal intervenes

$$\mathrm{T}_{[p\phi]} \ldots \mathbf{D}_{[\phi]} \ldots \mathrm{D}_{[g\phi]}$$

- Invisibility: hypothetical goal does not appear on tier

$$\mathrm{T}_{[p\phi]} \ldots \text{there} \ldots \mathrm{D}_{[g\phi]}$$

- Blocking: some non-agreeing element intervenes on the tier

$$\mathrm{T}_{[p\phi]} \ldots \mathbf{C} \ldots \mathrm{D}_{[g\phi]}$$

# Importance of the finite window

- Neither tiers nor the finite window alone are adequate.
  - Tiers allow long-distance dependencies to be treated as if local.
  - The finite constraint window limits the power of the system.
  - Together, they create the right type of relativized locality.

See Appendix 3 and Appendix 4 for details.

# Typological variation

# Parameters for variation

The parameters for TSL-2 (tier elements and constraints) correspond neatly to variation in long-distance dependencies.

- Visibility — which elements are relevant and which are ignored?
  - Case-sensitive agreement (cf. Bobaljik 2008; Preminger 2014)
- Iteration — if you allow AB and BB, then you get ABB, ABBB, etc.
  - Case/gender/number concord
- Directionality — do we ban AB or BA?
  - Upward/downward agreement (cf. Chomsky 2000; Zeijlstra 2012)

# Case-sensitive agreement

In Hindi, the verb agrees with the closest *nominative* argument, which may not be the subject.

(1) Hindi verbal agreement ignores ergatives (Mahajan 1990)

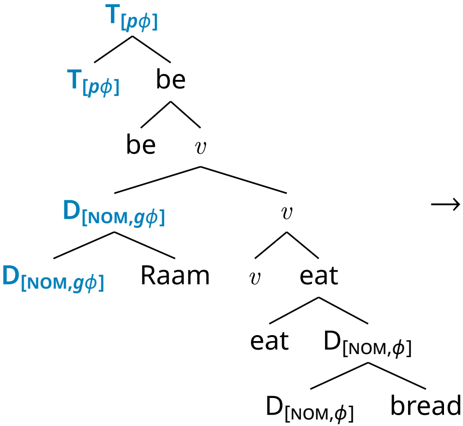
a. Raam roTii khaataa thaa.
Raam.M.**NOM** bread.F.NOM eat.IPFV.M be.PST.M
'Raam ate bread (habitually).'

b. Raam-ne roTii khaayii.
Raam.M-**ERG** bread.F.NOM eat.PFV.F
'Raam ate bread.'

**Analysis:** Project D only if nominative. Tier constraints are unchanged.

# Case-sensitive agreement (2)

‘Raam ate bread (habitually).’ (Nominative subject, subject agrees)




$$\rightarrow \quad \mathrm{T}_{[p\phi]} \cdot \mathrm{be} \cdot v \cdot \mathrm{D}_{[\mathrm{NOM},g\phi]} \cdot \mathrm{eat} \cdot \mathrm{D}_{[\mathrm{NOM},\phi]} \cdot \mathrm{bread}$$

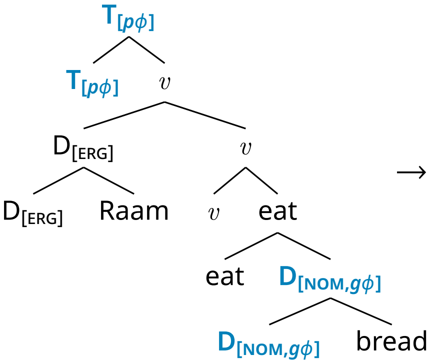
$$\downarrow$$

$$\mathrm{T}_{[p\phi]} \cdot \mathrm{D}_{[\mathrm{NOM},g\phi]} \cdot \mathrm{D}_{[\mathrm{NOM},\phi]}$$

We ignore agreement on the non-finite verb for simplicity. Concord will be discussed later.

# Case-sensitive agreement (3)

‘Raam ate bread.’ (Ergative subject, object agrees)




$$\rightarrow \mathbf{T}_{[p\phi]} \cdot v \cdot \mathrm{D}_{[\mathrm{ERG}]} \cdot \text{eat} \cdot \mathbf{D}_{[\mathrm{NOM},g\phi]} \cdot \text{bread}$$

$$\downarrow$$

$$\mathbf{T}_{[p\phi]} \cdot \mathbf{D}_{[\mathrm{NOM},g\phi]}$$

# Ergative ≠ Invisible

Ergatives are not invisible in Nepali (though datives are).

(2) Agreement with ergative in Nepali (Coon and Parker 2019)

a. Maile yas pasal-mā patrikaā kin-ē.
**1SG.ERG** DEM store-LOC newspaper.ABS buy-**1SG**
'I bought the newspaper in this store.'

b. Ma thag-ī-ē.
**1SG.ABS** cheat-PASS-**1SG**
'I was cheated.'

# Ergative ≠ Invisible

Ergatives are not invisible in Nepali (though datives are).

(3) Agreement with ergative in Nepali (Coon and Parker 2019)

a. Maile yas pasal-mā patrikaā kin-ē.
   1SG.ERG DEM store-LOC newspaper.ABS buy-1SG
   ‘I bought the newspaper in this store.’

b. Ma thag-ī-ē.
   1SG.ABS cheat-PASS-1SG
   ‘I was cheated.’

No problem! We project $D_{[NOM]}$ and $D_{[ERG]}$ but not $D_{[DAT]}$.

# Formal vs substantive constraints

- Bobaljik's (2008) case visibility hierarchy: Nom > Acc/Erg > Obliques
- We can encode the attested patterns in a TSL-2 grammar, but the implicational hierarchy itself requires a separate explanation.

# Concord in the DP

To allow for iterated agreement, just permit $p\phi \cdot p\phi$.

(4) Gender concord in German

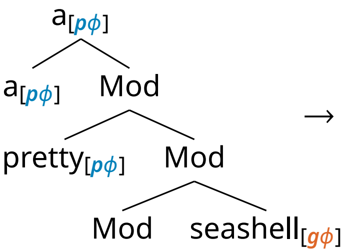
Ich habe [eine hübsche Muschel] gefunden.
I have [a.**F** pretty.**F** seashell.**F**] found

'I found a pretty seashell.'

**Analysis:** Ignore Mod on the tier, permit $\mathrm{D}_{[p\phi]} \cdot \mathrm{A}_{[p\phi]}$ and $\mathrm{A}_{[p\phi]} \cdot \mathrm{A}_{[p\phi]}$.

# Concord in the DP (2)

**Analysis:** Ignore Mod on the tier, permit $\mathrm{D}_{[p\phi]} \cdot \mathrm{A}_{[p\phi]}$ and $\mathrm{A}_{[p\phi]} \cdot \mathrm{A}_{[p\phi]}$.




$$\rightarrow \quad \mathrm{D}_{[p\phi]} \cdot \mathrm{Mod} \cdot \mathrm{A}_{[p\phi]} \cdot \mathrm{N}_{[g\phi]}$$

$$\downarrow$$

$$\mathrm{D}_{[p\phi]} \cdot \mathrm{A}_{[p\phi]} \cdot \mathrm{N}_{[g\phi]}$$

The Mod head is not crucial. If direct adjunction is used, then the pattern is local: the tier contains everything.

# Upward agreement

If the constraints are mirrored, then the direction of agreement is reversed.

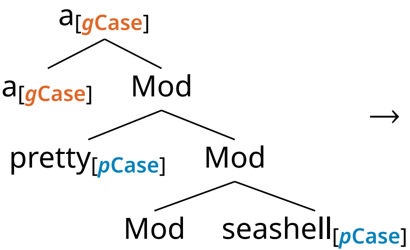
(5) Case concord in German

| | | | | | |
|---|---|---|---|---|---|
| Ich | habe | [eine | hübsche | Muschel] | gefunden. |
| I | have | [a.ACC | pretty.ACC | seashell.ACC] | found |

**Analysis:** allow $\mathrm{D}_{[g\mathrm{Case}]} \cdot \mathrm{A}_{[p\mathrm{Case}]}$ instead of $\mathrm{D}_{[p\mathrm{Case}]} \cdot \mathrm{A}_{[g\mathrm{Case}]}$, etc.

# Upward agreement (2)

**Analysis:** allow $\mathrm{D}_{[g\mathrm{Case}]} \cdot \mathrm{A}_{[p\mathrm{Case}]}$ instead of $\mathrm{D}_{[p\mathrm{Case}]} \cdot \mathrm{A}_{[g\mathrm{Case}]}$, etc.




$$\longrightarrow \quad \mathrm{D}_{[g\mathrm{Case}]} \cdot \mathrm{Mod} \cdot \mathrm{A}_{[p\mathrm{Case}]} \cdot \mathrm{N}_{[p\mathrm{Case}]}$$

$$\downarrow$$

$$\mathrm{D}_{[g\mathrm{Case}]} \cdot \mathrm{A}_{[p\mathrm{Case}]} \cdot \mathrm{N}_{[p\mathrm{Case}]}$$

We can handle definiteness agreement on the adjective (ignored here) in the same way.

# What does it mean to probe upward?

- In the MG derivation tree formalism (Graf and Shafiei 2019), we have a static representation of the entire derivation, so there is no problem.
- In a bottom-up Minimalist derivation, it is not obvious what it means for a probe to search upward. Some possibilities:
  - Let valued features search downward for unvalued features (Adger 2003)
  - Replace the search metaphor with the sliding window metaphor

# Typological variation – summary

| Example | Tier Elements | Tier Constraints |
|---|---|---|

# Typological variation – summary

| Example | Tier Elements | Tier Constraints |
|---|---|---|
| (Canonical) subject-verb agreement | All T/D/C | Strict pairing of $p\phi$ and $g\phi$ |

# Typological variation – summary

| Example | Tier Elements | Tier Constraints |
|---|---|---|
| (Canonical) subject-verb agreement | All T/D/C | Strict pairing of $p\phi$ and $g\phi$ |
| Case-sensitive agreement | All T/C<br>D only if right case | (as above) |
| Concord within DP | All D/Adj/N | Allow sequential $p\phi$ |

# Typological variation – summary

| Example | Tier Elements | Tier Constraints |
| --- | --- | --- |
| (Canonical) subject-verb agreement | All T/D/C | Strict pairing of $p\phi$ and $g\phi$ |
| Case-sensitive agreement | All T/C<br>D only if right case | (as above) |
| Concord within DP | All D/Adj/N | Allow sequential $p\phi$ |
| Upward agreement | (as above) | Swap order of $p\phi/g\phi$ |

# Parallels with phonology

| Parameter |
| --- |
| Participants |
| Invisible |
| Blockers |
| Directionality |
| Chaining |

See McMullin (2016) and McMullin and Hansson (2016) regarding long-distance harmony.

# Parallels with phonology

| Parameter | $\phi$-agreement |
|---|---|
| Participants | Probe and most DPs |
| Invisible | Non-DPs, some DPs |
| Blockers | Finite C, some DPs |
| Directionality | Downward/upward |
| Chaining | Concord/no concord |

See McMullin (2016) and McMullin and Hansson (2016) regarding long-distance harmony.

# Parallels with phonology

| Parameter | $\phi$-agreement | Vowel harmony |
|---|---|---|
| Participants | Probe and most DPs | Most vowels |
| Invisible | Non-DPs, some DPs | Consonants, some vowels |
| Blockers | Finite C, some DPs | Some vowels |
| Directionality | Downward/upward | Progressive/regressive |
| Chaining | Concord/no concord | Spreading/“icy targets” |

See McMullin (2016) and McMullin and Hansson (2016) regarding long-distance harmony.

# What else is TSL?

| Phenomenon | One line summary |
|---|---|
| Defective intervention* | Some DPs project even if they are never $g\phi$ |
| Probe horizons (Keine 2019) | V/$v$/T/C project even if they are never $p\phi$ |
| A′-agreement (Van Urk 2015)* | Only project DPs with a certain A′ feature |
| Omnivorous number | Only project DPs with [PL], not [SG] |
| Upward C agr. (Diercks 2013)* | C probes up, only project DPs that EPP-move |
| Default agreement* | Allow lone $p\phi$ under limited circumstances |
| Interaction/Satisfaction (Deal 2015)* | Allow multiple $g\phi$ under limited circumstances |
| Independent subfeatures of $\phi$ | Each probe gets its own tier/constraints |

Also: many movement (Graf 2022b) and case patterns (Vu et al. 2019; Hanson 2023b), though these analyses use a different tier-based model.

*See Hanson (2023a) and Hanson (2024a) for details.

# What *isn't* TSL?

Not all linguistic patterns are TSL. Of those that are not, most appear to be SS-TSL (structure-sensitive TSL). These include:

- Some long-distance harmony (De Santo and Graf 2019; Graf and Mayer 2018)
- Some tone patterns (e.g. unbounded tone plateauing)
- Some binding rules (Graf and Shafiei 2019)

# Strengths and limitations of the model

# Advantages of the model

- Clear separation of concerns:
  - Structural representation
  - Computations over said structure
  - Substance of elements of structure

# Advantages of the model

- Clear separation of concerns:
  - Structural representation
  - Computations over said structure
  - Substance of elements of structure
- Insights:
  - Agreement is especially similar to harmony as both involve feature matching; the same seems to be true of movement
  - If case is different, this is plausibly because it involves different kinds of constraints (e.g. dependent case)

# Limitations of the model

Puzzles for the path-based approach:

- What to do about violations of c-command (e.g. sub-command)?
- How to handle exceptions to the complement spine generalization?

# Limitations of the model

Puzzles for the path-based approach:

- What to do about violations of c-command (e.g. sub-command)?
- How to handle exceptions to the complement spine generalization?

What the TSL model (alone) does not tell us:

- Why does case matter for $\phi$-agreement? Why should nominatives always be visible, ergatives sometimes visible, and datives usually invisible?
- Why do probes seem to look downward more often than upward?
- How do children identify the visible elements and constraints for each dependency? (see Hanson 2024b; Belth 2023)

# Summary

- Agreement patterns in syntax are largely TSL with a window size of 2.

# Summary

- Agreement patterns in syntax are largely TSL with a window size of 2.
- If we vary the tier projection and constraints slightly, we can account for a wide range of variation across languages and constructions.

# Summary

- Agreement patterns in syntax are largely TSL with a window size of 2.
- If we vary the tier projection and constraints slightly, we can account for a wide range of variation across languages and constructions.
- This variation is similar to other linguistic phenomena, especially phonological harmony.

# Summary

- Agreement patterns in syntax are largely TSL with a window size of 2.
- If we vary the tier projection and constraints slightly, we can account for a wide range of variation across languages and constructions.
- This variation is similar to other linguistic phenomena, especially phonological harmony.
- Most of the logical possibilities of the model are realized within a single phenomenon — this is not necessarily expected!

# Some open questions

- Do we ever need a window size larger than 2?

# Some open questions

- Do we ever need a window size larger than 2?
- Are there patterns that are not TSL under any reasonable analysis?

# Some open questions

- Do we ever need a window size larger than 2?
- Are there patterns that are not TSL under any reasonable analysis?
- How far can we take the parallel with harmony in phonology?

# Takeaways

- Computational approaches to linguistic analysis reveal insights that might otherwise not be obvious.

# Takeaways

- Computational approaches to linguistic analysis reveal insights that might otherwise not be obvious.
- In other cases, they provide independent support to conclusions reached in other ways (e.g. visibility is parameterized).

# Takeaways

- Computational approaches to linguistic analysis reveal insights that might otherwise not be obvious.
- In other cases, they provide independent support to conclusions reached in other ways (e.g. visibility is parameterized).
- A clear understanding of the formal patterns can help us understand other aspects of linguistic structure.

# Acknowledgments

This work was partly supported by NSF Grant BCS-1845344 and by the Institute for Advanced Computational Science at Stony Brook University.

INSTITUTE FOR ADVANCED COMPUTATIONAL SCIENCE

Thanks to Thomas Graf, Sandhya Sundaresan, Tom McFadden, and John Bailyn for comments and feedback throughout the project.

# References


Adger, David (2003). *Core syntax: A minimalist approach*. Vol. 20.

Belth, Caleb (2023). "Towards a Learning-Based Account of Underlying Forms: A Case Study in Turkish". In: *Proceedings of the Society for Computation in Linguistics 2023*.

Bobaljik, Jonathan (2008). "Where's Phi? Agreement as a Postsyntactic Operation". In: *Phi Theory*.

Brody, Michael (2000). "Mirror theory: Syntactic representation in perfect syntax". In: *Linguistic Inquiry* 31.1.

Chomsky, Noam (2000). "Minimalist inquiries: The framework". In: *Step by Step: Essays on Minimalist Syntax in Honor of Howard Lasnik*. First Published in MIT Occasional Papers in Linguistics 15, 1998.

Coon, Jessica and Clint Parker (2019). "Case Interactions in Syntax". In: *Oxford Research Encyclopedia of Linguistics*.

De Santo, Aniello and Thomas Graf (2019). "Structure Sensitive Tier Projection: Applications and Formal Properties". In: *Formal Grammar*.

Deal, Amy Rose (2015). "Interaction and satisfaction in $\phi$-agreement". In: *Proceedings of NELS 45*.

Diercks, Michael (2013). "Indirect agree in Lubukusu complementizer agreement". In: *Natural Language & Linguistic Theory* 31.2.

Frank, Rober and K. Vijay-Shankar (2001). "Primitive c-command". In: *Syntax*.

# References (2)


Goldsmith, John (1976). “Autosegmental phonology”. PhD thesis. Massachusetts Institute of Technology.

Graf, Thomas (2022a). “Subregular linguistics: bridging theoretical linguistics and formal grammar”. In: *Theoretical Linguistics* 48.3–4.

— (2022b). “Typological implications of tier-based strictly local movement”. In: *Proceedings of the Society for Computation in Linguistics 2022*.

Graf, Thomas and Aniello De Santo (2019). “Sensing Tree Automata as a Model of Syntactic Dependencies”. In: *Proceedings of the 16th Meeting on the Mathematics of Language*.

Graf, Thomas and Kalina Kostyszyn (2021). “Multiple Wh-Movement is not Special: The Subregular Complexity of Persistent Features in Minimalist Grammars”. In: *Proceedings of the Society for Computation in Linguistics 2021*.

Graf, Thomas and Connor Mayer (2018). “Sanskrit n-Retroflexion is Input-Output Tier-Based Strictly Local”. In: *Proceedings of SIGMORPHON 2018*.

Graf, Thomas and Nazila Shafiei (2019). “C-command dependencies as TSL string constraints”. In: *Proceedings of the Society for Computation in Linguistics 2019*.

Hanson, Kenneth (2023a). *A Computational Perspective on the Typology of Agreement*. Oral presentation. NYU Linguistics Brown Bag.

— (2023b). “A TSL Analysis of Japanese Case”. In: *Proceedings of the Society for Computation in Linguistics 2023*.

— (2024a). “Tier-Based Strict Locality and the Typology of Agreement”. Under review.

# References (3)


Hanson, Kenneth (2024b). “Tiers, Paths, and Syntactic Locality: The View from Learning”. To appear in Proceedings of SCiL 2024.

Heinz, Jeffrey et al. (2011). “Tier-based strictly local constraints for phonology”. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human language technologies*.

Keine, Stefan (2019). “Selective Opacity”. In: *Linguistic Inquiry* 50.1.

Lambert, Dakotah et al. (2021). “Typology emerges from simplicity in representations and learning”. In: *Journal of Language Modelling* 9.1.

Mahajan, Anoop Kumar (1990). “The A/A-bar distinction and movement theory”. PhD thesis. Massachusetts Institute of Technology.

McMullin, Kevin (2016). “Tier-based locality in long-distance phonotactics: learnability and typology”. PhD thesis. University of British Columbia.

McMullin, Kevin and Gunnar Ólafur Hansson (2016). “Long-Distance Phonotactics as Tier-Based Strictly 2-Local Languages”. In: *Proceedings of the Annual Meetings on Phonology*. Vol. 2.

Preminger, Omer (2014). *Agreement and its failures*. Vol. 68.

Van Urk, Coppe (2015). “A Uniform Syntax for Phrasal Movement: A Dinka Bor Case Study”. PhD thesis. Massachusetts Institute of Technology.

Vu, Mai Ha et al. (2019). “Case assignment in TSL syntax: A case study”. In: *Proceedings of the Society for Computation in Linguistics 2019*.

Zeijlstra, Hedde (2012). “There is only one way to agree”. In: *The Linguistic Review* 29.3.

## Extras

Even more on TSL

Some formal details

More on locality

Computational considerations

# Extra example: Sibilant harmony

Sibilants match in anteriority, *t* blocks harmony, other C's transparent

(based on Slovenian)

| | |
|---|---|
| All elements: | {s, ʃ, t, k, a} |
| Tier elements: | {s, ʃ, t} |
| Constraints: | {*sʃ, *ʃs} |

| Word | Tier | |
|---|---|---|
| s a s a s a | s s s | ✓ |
| s a **s** a **ʃ** a | s **s** **ʃ** | ✗ |
| s a k a s a | s s | ✓ |
| **s** a k a **ʃ** a | **s** **ʃ** | ✗ |
| s a t a s a | s t s | ✓ |
| s a t a ʃ a | s t ʃ | ✓ |

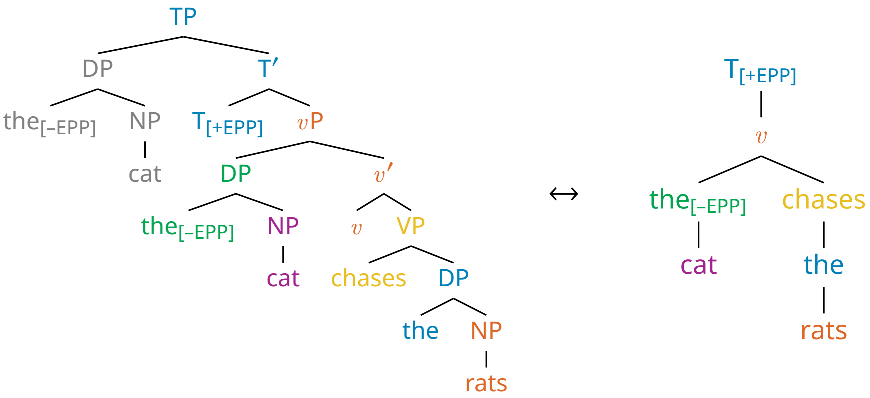
# TSL string languages – formal definition

In a **tier-based strictly *k*-local (TSL-*k*)** language, a string is well-formed iff its **tier projection** does not contain any forbidden substrings of some length $k$.

- $\Sigma$ = "alphabet" = set of all symbols
- $T$ = "tier alphabet" = set of visible symbols
- $G$ = "grammar" = forbidden substrings
- The tier projection is obtained by deleting all non-tier elements and concatenating the remaining elements.

# MG derivation trees

- All nodes appear in base position.
- The rightmost child of a node is its complement; others are specifiers.
- Movement is indicated using feature diacritics.




See Graf and Kostyszyn (2021) for a formal definition. Related: Brody (2000).

# Command strings

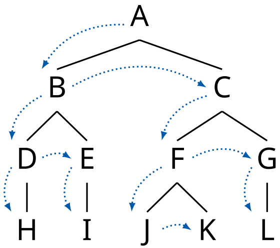
A **command string** (c-string) is a derivational ordering of nodes.

- There is a c-string from the root to each node.
- Among each head and its arguments: Head < Specifier < Complement.

# Command strings

A **command string** (c-string) is a derivational ordering of nodes.

- There is a c-string from the root to each node.
- Among each head and its arguments: Head < Specifier < Complement.

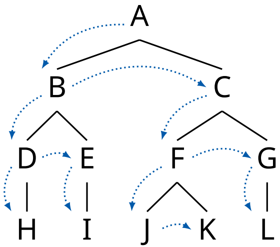


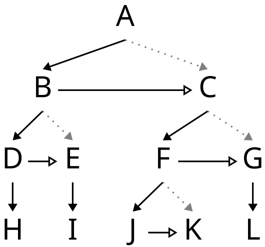

See Graf and Shafiei (2019) for details.

# Command strings

A **command string** (c-string) is a derivational ordering of nodes.

- There is a c-string from the root to each node.
- Among each head and its arguments: Head < Specifier < Complement.

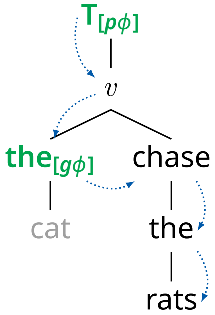



See Graf and Shafiei (2019) for details.

# Tiers over command strings

✓ The cat **chases** the rats. (subject agreement)



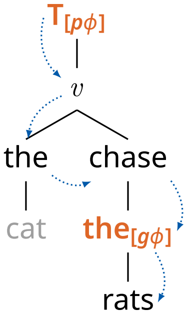

$\longrightarrow$ $\mathbf{T}_{[p\phi]} \cdot v \cdot \mathbf{the}_{[g\phi]} \cdot \text{chase} \cdot \text{the} \cdot \text{rats}$

$\downarrow$

$\mathbf{T}_{[p\phi]} \cdot \mathbf{the}_{[g\phi]} \cdot \text{the}$

# Tiers over command strings (2)

✗ The cat **chase** the rats. (object agreement)



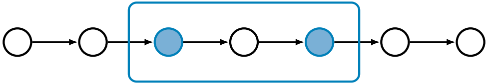

$\longrightarrow$ $\mathrm{T}_{[p\phi]} \cdot v \cdot \mathrm{the} \cdot \mathrm{chase} \cdot \mathrm{the}_{[g\phi]} \cdot \mathrm{rats}$

$\downarrow$

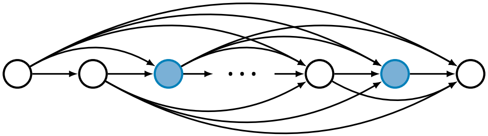
$\mathrm{T}_{[p\phi]} \cdot \mathrm{the} \cdot \mathrm{the}_{[g\phi]}$

# Three models of locality

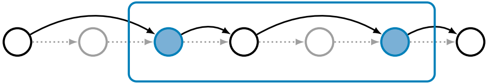
## Immediate precedence (SL)



## General precedence (SP)



## Tier precedence (TSL)



# Three models of locality (2)

- The immediate precedence (SL) model can handle local spreading.
- The general precedence (SP) model can handle unbounded processes, but can't handle blockers.
- Only the tier precedence (TSL) model can handle unbounded processes with blocking.

# Limits on structural configurations

TSL computations can relate elements at a distance, but are otherwise severely restricted in what they can do.

- No arbitrary logic — “a DP can A-move out of a finite CP, but only if there is A′-movement within some (other) CP in the sentence”
- No counting — “up to three reflexive pronouns may occur in a sentence if each obeys the Binding Theory”

# Limits on structural configurations

TSL computations can relate elements at a distance, but are otherwise severely restricted in what they can do.

- No arbitrary logic — “a DP can A-move out of a finite CP, but only if there is A′-movement within some (other) CP in the sentence”
- No counting — “up to three reflexive pronouns may occur in a sentence if each obeys the Binding Theory”

These characteristics derive from the restriction that all constraints must be stated within the moving window.

# Conditions for efficient learning

- The restrictions on TSL patterns help to make them efficiently learnable by limiting the amount of memory needed (Lambert et al. 2021).
- But there are too many possible tiers to test them all individually.
- We also need to consider other aspects of language acquisition such as the Tolerance Principle (see eg. Belth 2023; Hanson 2024b).