Do we Need Features for Morphosyntax?

Thomas Graf

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

ZAS
Jun 26, 2017
Two Routes Towards Generalizations

Why Route 2?

- Many Surface to Deep mappings
- Systematize first, then implement at Deep level
Two Routes Towards Generalizations

- Deep
- Surface
- Description
- Pattern
- System

Why Route 2?

- Many Surface to Deep mappings
- Systematize first, then implement at Deep level
Two Routes Towards Generalizations

Why Route 2?

- Many Surface to Deep mappings
- Systematize first, then implement at Deep level
Two Routes Towards Generalizations

Why Route 2?

- Many Surface to Deep mappings
- Systematize first, then implement at Deep level
Two Routes Towards Generalizations

Why Route 2?

- Many Surface to Deep mappings
- Systematize first, then implement at Deep level
Two Routes Towards Generalizations

Why Route 2?

- Many Surface to Deep mappings
- Systematize first, then implement at Deep level
A Case Study: *ABA and PCC

*ABA Generalization (Bobaljik 2012)

Two paradigmatic cells cannot be syncretic to the exclusion of any intervening cell.

(1) a. smart, smarter, smartest (AAA)
b. good, better, best (ABB)
c. *good, better, goodest (ABA)

Person Case Constraint (PCC; Bonet 1994; Walkow 2012)

The well-formedness of clitic combinations is contingent on their person specification.

(2) Roger le/*me leur a présenté.
Roger 3SG.ACC/1SG.ACC 3PL.DAT has shown
‘Roger has shown me/him to them.’
A Case Study: *ABA and PCC

*ABA Generalization (Bobaljik 2012)

Two paradigmatic cells cannot be syncretic to the exclusion of any intervening cell.

(1) a. smart, smarter, smartest (AAA)
    b. good, better, best (ABB)
    c. * good, better, goonest (ABA)

Person Case Constraint (PCC; Bonet 1994; Walkow 2012)

The well-formedness of clitic combinations is contingent on their person specification.

(2) Roger le/*me leur a présenté.
    Roger 3SG.ACC/1SG.ACC 3PL.DAT has shown
    ‘Roger has shown me/him to them.’
Outline

1. The *ABA Generalization: Monotonicity

2. *ABA Revisited: Graph-Theoretic Approach
   - Application to Pronoun Syncretism
   - Computational Motivation
   - Beyond 3-Cell Systems

3. Person Case Constraint
*ABA: A First Account

- **Syncretism**: multiple cells mapped to the same output
- A mapping that produces ABA violates **monotonicity**.

**Monotonicity for Pronoun Syncretism**

- Suppose $3 < 2 < 1$ (Zwicky 1977)
- A function $f$ is **monotonic** iff $x \leq y$ implies $f(x) \leq f(y)$.
- No monotonic function from $\{1, 2, 3\}$ to $\{A, B, C\}$ can produce ABA!
- This holds irrespective of the structure of $\{A, B, C\}$.
Illustrating Monotonicity

- Monotonicity is similar to No Crossing Branches constraint in autosegmental phonology. (Goldsmith 1976)

Patterns:

- But why should spell-out functions be monotonic?
Illustrating Monotonicity

- Monotonicity is similar to No Crossing Branches constraint in autosegmental phonology. (Goldsmith 1976)

\[
\begin{array}{c|c|c}
1 & 2 & 3 \\
A & B & C \\
\end{array}
\]

Patterns:

- But why should spell-out functions be monotonic?
Illustrating Monotonicity

- Monotonicity is similar to No Crossing Branches constraint in autosegmental phonology. (Goldsmith 1976)

Patterns: ABC,

- But why should spell-out functions be monotonic?
Illustrating Monotonicity

- Monotonicity is similar to No Crossing Branches constraint in autosegmental phonology. (Goldsmith 1976)

\[
\begin{array}{ccc}
1 & 2 & 3 \\
|    |   |  \\
A    & B & C
\end{array}
\]

Patterns: ABC,

- But why should spell-out functions be monotonic?
Monotonicity is similar to No Crossing Branches constraint in autosegmental phonology. (Goldsmith 1976)

Patterns: ABC, AAB = AAC,

But why should spell-out functions be monotonic?
Illustrating Monotonicity

- Monotonicity is similar to No Crossing Branches constraint in autosegmental phonology. (Goldsmith 1976)

Patterns: ABC, AAB = AAC,

- But why should spell-out functions be monotonic?
Illustrating Monotonicity

- Monotonicity is similar to No Crossing Branches constraint in autosegmental phonology. (Goldsmith 1976)

```
1  2  3
A   B   C
```

**Patterns:** ABC, AAB = AAC, ABB = ACC,

- But why should spell-out functions be monotonic?
Illustrating Monotonicity

Monotonicity is similar to No Crossing Branches constraint in autosegmental phonology. (Goldsmith 1976)

Patterns: ABC, AAB = AAC, ABB = ACC,

But why should spell-out functions be monotonic?
Illustrating Monotonicity

- Monotonicity is similar to No Crossing Branches constraint in autosegmental phonology. (Goldsmith 1976)

Patterns: ABC, AAB = AAC, ABB = ACC, ABC

- But why should spell-out functions be monotonic?
Monotonicity is similar to No Crossing Branches constraint in autosegmental phonology. (Goldsmith 1976)

Patterns: ABC, AAB = AAC, ABB = ACC, ABC

But why should spell-out functions be monotonic?
A More General View: Graph Structure Preservation

The General Idea

- *ABA is about structure preservation.
- Syncretism is modification of a base graph.
- Modification must not contradict orderings of base graph.

Definition (Weakly Non-Inverting Graph Mappings)

- Given input graph $G$ and output graph $G'$
  - $x \triangleleft y$ iff $y$ is reachable from $x$ in $G$,
  - $x \blacktriangleright y$ iff $y$ is reachable from $x$ in $G'$.
- A mapping from $G$ to $G'$ is weakly non-inverting iff
  $x \triangleleft y \land y \blacktriangleright x \rightarrow x \blacktriangleright y$
Weakly Non-Inverting Graph Mappings

- Since we want graphs to encode hierarchies, they must be *weakly connected*: ignoring the direction of arrows, all nodes are mutually reachable.
- And the mapping must be weakly non-inverting:
  \[ x \triangleleft y \land y \triangleright x \rightarrow x \triangleright y \]
Since we want graphs to encode hierarchies, they must be *weakly connected*: ignoring the direction of arrows, all nodes are mutually reachable.

And the mapping must be weakly non-inverting:

$$x \triangleleft y \land y \triangleright x \rightarrow x \triangleright y$$
Weakly Non-Inverting Graph Mappings

- Since we want graphs to encode hierarchies, they must be *weakly connected*: ignoring the direction of arrows, all nodes are mutually reachable.
- And the mapping must be weakly non-inverting: \( x ◁ y \land y △ x \rightarrow x △ y \)
Weakly Non-Inverting Graph Mappings

- Since we want graphs to encode hierarchies, they must be *weakly connected*: ignoring the direction of arrows, all nodes are mutually reachable.
- And the mapping must be weakly non-inverting: 
  \[ x \triangleleft y \land y \triangleleft x \rightarrow x \triangleleft y \]
Weakly Non-Inverting Graph Mappings

- Since we want graphs to encode hierarchies, they must be *weakly connected*: ignoring the direction of arrows, all nodes are mutually reachable.

- And the mapping must be weakly non-inverting:
  \[ x \triangleleft y \land y \triangleright x \rightarrow x \triangleright y \]
Weakly Non-Inverting Graph Mappings

- Since we want graphs to encode hierarchies, they must be weakly connected: ignoring the direction of arrows, all nodes are mutually reachable.

- And the mapping must be weakly non-inverting:

  \[ x \triangleleft y \land y \blacktriangleleft x \rightarrow x \blacktriangleleft y \]
Weakly Non-Inverting Graph Mappings

- Since we want graphs to encode hierarchies, they must be *weakly connected*: ignoring the direction of arrows, all nodes are mutually reachable.
- And the mapping must be weakly non-inverting:
  \[
  x \triangleleft y \land y \triangleright x \rightarrow x \triangleright y
  \]
Since we want graphs to encode hierarchies, they must be weakly connected: ignoring the direction of arrows, all nodes are mutually reachable.

And the mapping must be weakly non-inverting:

\[ x \triangleleft y \land y \triangleright x \rightarrow x \triangleright y \]
Weakly Non-Inverting Graph Mappings

- Since we want graphs to encode hierarchies, they must be weakly connected: ignoring the direction of arrows, all nodes are mutually reachable.
- And the mapping must be weakly non-inverting:
  \[ x \triangleleft y \land y \triangleright x \rightarrow x \triangleright y \]
Since we want graphs to encode hierarchies, they must be weakly connected: ignoring the direction of arrows, all nodes are mutually reachable.

And the mapping must be weakly non-inverting:

\[ x \preceq y \land y \prec x \rightarrow x \preceq y \]
Weakly Non-Inverting Graph Mappings

- Since we want graphs to encode hierarchies, they must be *weakly connected*: ignoring the direction of arrows, all nodes are mutually reachable.
- And the mapping must be weakly non-inverting:
  \[ x \triangleleft y \land y \rightleftharpoons x \rightarrow x \rightleftharpoons y \]
Weakly Non-Inverting Graph Mappings

- Since we want graphs to encode hierarchies, they must be **weakly connected**: ignoring the direction of arrows, all nodes are mutually reachable.

- And the mapping must be weakly non-inverting:
  \[ x \triangleleft y \land y \triangleright x \rightarrow x \triangleright y \]
Weakly Non-Inverting Graph Mappings

- Since we want graphs to encode hierarchies, they must be *weakly connected*: ignoring the direction of arrows, all nodes are mutually reachable.
- And the mapping must be weakly non-inverting:
  \[ x \triangleleft y \land y \triangleright x \rightarrow x \triangleleft y \]
Weakly Non-Inverting Graph Mappings

- Since we want graphs to encode hierarchies, they must be *weakly connected*: ignoring the direction of arrows, all nodes are mutually reachable.
- And the mapping must be weakly non-inverting:
  \[ x \triangleleft y \land y \triangleright x \rightarrow x \triangleright y \]
Weakly Non-Inverting Graph Mappings

- Since we want graphs to encode hierarchies, they must be weakly connected: ignoring the direction of arrows, all nodes are mutually reachable.
- And the mapping must be weakly non-inverting:
  \[ x \triangleleft y \land y \triangleright x \rightarrow x \triangleright y \]
Weakly Non-Inverting Graph Mappings

- Since we want graphs to encode hierarchies, they must be *weakly connected*: ignoring the direction of arrows, all nodes are mutually reachable.

- And the mapping must be weakly non-inverting:
  \[ x \triangleleft y \land y \triangleleft x \rightarrow x \triangleleft y \]
Graphs and Syncretism

- Suppose two cells may be syncretic iff they are mutually reachable in a graph.
- Then the previous set of graphs describes the class of attested syncretisms.
Graphs and Syncretism

- Suppose two cells may be syncretic iff they are mutually reachable in a graph.
- Then the previous set of graphs describes the **class of attested syncretisms**.
Graphs and Syncretism

- Suppose two cells may be syncretic iff they are mutually reachable in a graph.
- Then the previous set of graphs describes the **class of attested syncretisms**.
Graphs and Syncretism

- Suppose two cells may be syncretic iff they are mutually reachable in a graph.
- Then the previous set of graphs describes the **class of attested syncretisms**.
Graphs and Syncretism

▶ Suppose two cells may be syncretic iff they are mutually reachable in a graph.
▶ Then the previous set of graphs describes the **class of attested syncretisms**.
Graphs and Syncretism

▶ Suppose two cells may be syncretic iff they are mutually reachable in a graph.
▶ Then the previous set of graphs describes the class of attested syncretisms.
Suppose two cells may be syncretic iff they are mutually reachable in a graph. Then the previous set of graphs describes the class of attested syncretisms.
Why Weakly Non-Inverting Maps?

- The restriction to weakly non-inverting maps reduces computational complexity.
- These graph mappings correspond to strictly 1-local string mappings.
- Those are the weakest class of mappings.
- So the *ABA generalization has a third-factor explanation: (Chomsky 2005)
  - independent base hierarchy of cells
  - computationally limited changes to hierarchy
Some morphosyntactic phenomena have many different cells. 
case syncretism, noun stem allomorphy

Those do not scale well for feature combinatorics.

Weakly non-inverting maps still obey \( *ABA \) 
if output graphs must be connected:

\[
\forall x, y [ x \leftarrow y \lor y \leftarrow x ]
\]

Weakly non-inverting + strong connectedness = 
base arrows must not be removed
Case Syncretism

- Modified case hierarchy as base
  (Blake 2001)
- Allows syncretism of both
  Acc & Dat and Acc & Gen
  (Harðarson 2016)
Interim Summary

- Weakly non-inverting graph mappings preserve aspects of the base order.
- This structure preservation derives the *ABA generalization.
- Some ad hoc stipulations are still needed in certain cases.
- Those reflect aspects of the syntactic mechanisms, which the graph-theoretic view abstracts away from.

<table>
<thead>
<tr>
<th>Phenomenon</th>
<th>Target graph</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pronoun allomorphy</td>
<td>(weakly) connected</td>
<td>none</td>
</tr>
<tr>
<td>Adjectival gradation</td>
<td>(weakly) connected</td>
<td>$2 \triangleleft 1 \to 3 \triangleleft 1$</td>
</tr>
<tr>
<td>Case syncretism</td>
<td>connected</td>
<td>none</td>
</tr>
<tr>
<td>Noun stem suppletion</td>
<td>connected</td>
<td>$\neg \exists z [z \triangleleft x] \to (y \triangleleft x \to x \triangleleft y)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\exists z [z \triangleleft x] \to (x \triangleleft y \leftrightarrow y \triangleleft x)$</td>
</tr>
</tbody>
</table>
There are four attested variants of the PCC:

- **S(strong)-PCC** DO must be 3.
  
  *(Bonet 1994)*

- **U(ltrastrong)-PCC** DO is less prominent than IO, where 3 is less prominent than 2, and 2 is less prominent than 1.
  
  *(Nevins 2007)*

- **W(eak)-PCC** 3IO combines only with 3DO.
  
  *(Bonet 1994)*

- **M(e first)-PCC** If IO is 2 or 3, then DO is not 1.
  
  *(Nevins 2007)*

- But symmetric variants have been discovered.
  
  *(Stegovec 2016)*

- This looks like a mess!
A More Systematic Perspective (Walkow 2012)

<table>
<thead>
<tr>
<th>IO↓/DO→</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NA</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>2</td>
<td>*</td>
<td>NA</td>
<td>✓</td>
</tr>
<tr>
<td>3</td>
<td>*</td>
<td>*</td>
<td>NA</td>
</tr>
</tbody>
</table>

**U-PCC**

<table>
<thead>
<tr>
<th>IO↓/DO→</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NA</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>2</td>
<td>✓</td>
<td>NA</td>
<td>✓</td>
</tr>
<tr>
<td>3</td>
<td>*</td>
<td>*</td>
<td>NA</td>
</tr>
</tbody>
</table>

**W-PCC**

<table>
<thead>
<tr>
<th>IO↓/DO→</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NA</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>2</td>
<td>✓</td>
<td>NA</td>
<td>✓</td>
</tr>
<tr>
<td>3</td>
<td>*</td>
<td>*</td>
<td>NA</td>
</tr>
</tbody>
</table>

**S-PCC**

<table>
<thead>
<tr>
<th>IO↓/DO→</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NA</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>2</td>
<td>*</td>
<td>NA</td>
<td>✓</td>
</tr>
<tr>
<td>3</td>
<td>*</td>
<td>✓</td>
<td>NA</td>
</tr>
</tbody>
</table>

**M1-PCC**
Graph-Theoretic Unification

**Generalized PCC**
y must not be reachable from \( x \).

**Standard PCCs:**
y = IO, \( x = DO \)

**Symmetric PCCs:**
y = DO, \( x = IO \)
### Overview of Relevant Graph Classes

<table>
<thead>
<tr>
<th>Phenomenon</th>
<th>Target graph</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pronoun allomorphy</td>
<td>(w-)connected</td>
<td>none</td>
</tr>
<tr>
<td>Adjectival gradation</td>
<td>(w-)connected</td>
<td>2 ▶ 1 → 3 ▶ 1</td>
</tr>
<tr>
<td>Case syncretism</td>
<td>connected</td>
<td>none</td>
</tr>
<tr>
<td>Noun stem suppletion</td>
<td>connected</td>
<td>(z ◀ x) → (y ▶ x → x ▶ y)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(z ◀ x) → (x ▶ y ↔ y ▶ x)</td>
</tr>
<tr>
<td>PCC</td>
<td>w-connected</td>
<td>(z ◀ x) → (y ▶ x → x ▶ y)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(x ◀ z) → ¬∃z[x ◀ z]</td>
</tr>
</tbody>
</table>
Conclusion

- Graphs generalize across domains of morphosyntax
- No need for features, talk directly about cells
- Scales better than combinatorics
- Can be a theory of markedness rather than well-formedness
- **But:** a lot of work still to be done
  Gender Case Constraint, inverse marking, resolved agreement, ...
References


