Dependencies in Syntax and Phonology: A Computational Comparison

Thomas Graf (and Jeff Heinz, sorta)
mail@thomasgraf.net
http://thomasgraf.net

Department of Linguistics
Stony Brook University

NECPHON
November 15 2014

Today's Topic

The Received View on Syntax and Phonology

- Little cross-talk between syntax and phonology
- Properties of one supposedly have no bearing on the other
- Huge difference with respect to weak generative capacity

Counter Position

- Computationally, phonology and syntax are very similar.
- Over linguistically plausible models, both rely on dependencies of the same computational complexity.
- Main difference is data structures: **strings** versus **trees**

Today's Topic

The Received View on Syntax and Phonology

- Little cross-talk between syntax and phonology
- Properties of one supposedly have no bearing on the other
- Huge difference with respect to weak generative capacity

Counter Position

- Computationally, phonology and syntax are very similar.
- Over linguistically plausible models, both rely on dependencies of the same computational complexity.
- Main difference is data structures: **strings** versus **trees**

Outline

Linguistics: Syntax and Phonology are Unrelated

- **Different Frameworks**

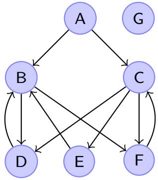
Aspects	⇔	SPE
GB	⇔	Autosegmental/GP
Minimalism	⇔	OT

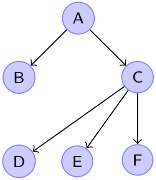
- **Empirical Separation**
  - What is the syntactic analog of umlaut or final devoicing?
  - What is the phonological analog of passive or the Person Case Constraint?
  - Cognitive impairments often impact only one of the two.

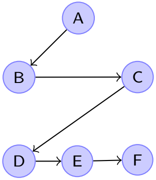
Language as Sets

Formally, a language is simply
a set of objects of a specific type:

- **graph**: structure of connected nodes
  flow chart, *street network*, *Wikipedia*, *internet*, *video game AI*
- **tree**: connected graph where every node is reachable from at most one node
  family tree, *hard drive layout*, *XML file*
- **string**: sequence of nodes
  telephone number, *Python program*, *human genome*, *Shakespeare's oeuvre*

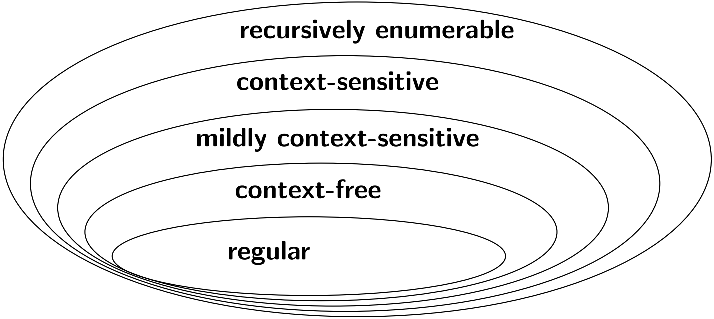






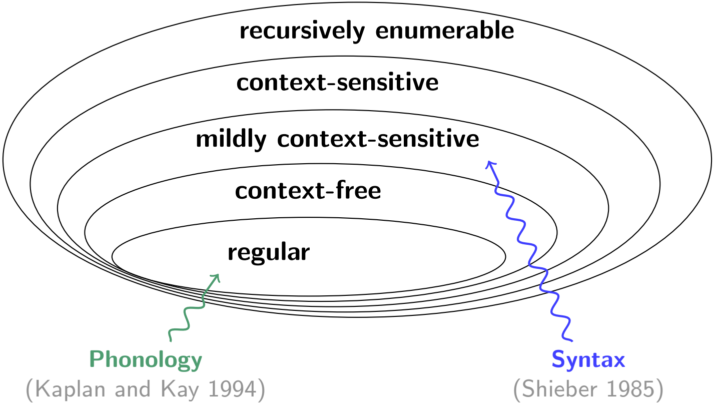
The Chomsky Hierarchy of String Languages

- The perceivable output of language is strings (sequences of sound waves, words, sentences).
- The complexity of string languages is measured by the (extended) **Chomsky hierarchy**. (Chomsky 1956, 1959)



The Chomsky Hierarchy of String Languages

- The perceivable output of language is strings (sequences of sound waves, words, sentences).
- The complexity of string languages is measured by the (extended) **Chomsky hierarchy**. (Chomsky 1956, 1959)



Pinpointing Phonology in the Chomsky Hierarchy

- Regular languages are too powerful for phonology.
- Jeff Heinz has argued that phonology can be described by a natural generalization of *n*-gram grammars. (Heinz et al. 2011)
- **Idea:** non-local dependencies are **local on phonological tiers**

(Negative) Bigram Grammars

- Suppose we have a fixed alphabet Σ (e.g. sounds of English).
- A bigram is a sequence **ab** s.t. **a**, **b** $\in \Sigma$.
- A **bigram grammar** G is a finite set of bigrams.
- G generates the largest language of strings such that no string contains any bigrams of G as a substring
- Intuition: **bigrams are hard, local constraints**

Example

Rewrite rule	Constraint	Bigrams
n $\rightarrow$ m |_ **b**	***nb**	**nb**
z $\rightarrow$ s |_ **$**	***z$**	**z$**
[-voice] $\rightarrow$ $\emptyset$ |_ **$**	***[-voice]$**	**s$**, **θ$**, **f$**, ...

(Negative) Bigram Grammars

- Suppose we have a fixed alphabet Σ (e.g. sounds of English).
- A bigram is a sequence **ab** s.t. **a**, **b** $\in \Sigma$.
- A **bigram grammar** G is a finite set of bigrams.
- G generates the largest language of strings such that no string contains any bigrams of G as a substring
- Intuition: **bigrams are hard, local constraints**

Example

Rewrite rule	Constraint	Bigrams
n $\rightarrow$ m | _ **b**	***nb**	**nb**
z $\rightarrow$ s | _ **$**	***z$**	**z$**
[-voice] $\rightarrow$ $\emptyset$ | _ **$**	***[-voice]$**	**s$**, **θ$**, **f$**, ...

Tiers for Long-Distance Dependencies

- We can move to 3-grams, 4-grams, ... n-grams in order to regulate less local processes (e.g. umlaut).
- **Problem:** Still limited to locality domain of size n
  $\Rightarrow$ unbounded processes cannot be captured
- **Solution:** segments can be on multiple tiers

Tier-Based Bigram Grammar

- Let $T \subseteq \Sigma$ be a tier alphabet.
- A **tier-based bigram grammar** G is a pair of finite sets of bigrams over Σ and T, respectively.
- G generates s iff
  - s has no Σ-bigram as a substring,
  - the restriction of s to elements of T has no T-bigram as a substring.

Tiers for Long-Distance Dependencies

- We can move to 3-grams, 4-grams, ... n-grams in order to regulate less local processes (e.g. umlaut).
- **Problem:** Still limited to locality domain of size n
  $\Rightarrow$ unbounded processes cannot be captured
- **Solution:** segments can be on multiple tiers

Tier-Based Bigram Grammar

- Let $T \subseteq \Sigma$ be a tier alphabet.
- A **tier-based bigram grammar** G is a pair of finite sets of bigrams over Σ and T, respectively.
- G generates s iff
  - s has no Σ-bigram as a substring,
  - the restriction of s to elements of T has no T-bigram as a substring.

Example: Sibilant Harmony

Rewrite rule	Constraint	Tier-Bigram
s → ʃ | ʃ . . . _	*ʃ . . . s	ʃs

T-Tier:	$		ʃ		s		$
Σ-Tier:	$	e	ʃ	i	s	i	$

Example: Primary Stress Assignment

- Every word must have exactly one primary stress.
- Let T be the set of symbols with primary stress.
- Then we need the following T-bigrams:
  - $$ "at least one primary stress"
  - ab "not more than one primary stress" (for all $a, b \in T$)

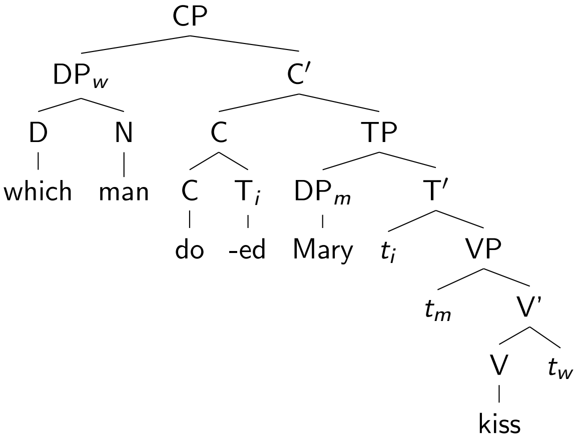
Stress Missing	**Too Many Stresses**	**Just Right**
$ $	$ 'e 'i $	$ 'e $
$ e ʃ i s i $	$ 'e ʃ 'i s i $	$ 'e ʃ i s i $

Phonology is Subregular

- Tier-based *n*-gram grammars generate only a subclass of the regular languages.
- Only a few known phenomena might not to be tier-local, but the data is unclear (e.g. primary stress assignment in Cairene Arabic; Graf 2010)
- Hence **phonology is subregular**.
- **Core insights**
  - Most phonological dependencies are local.
  - Non-local dependencies are local between the elements of the relevant type ($\approx$ tier).

A Closer Look at Syntax

The MCS-result treats syntactic patterns as string dependencies.
But **syntacticians work with trees**, not strings.



Minimalist Grammars

- **Minimalism** is the dominant syntactic theory. (Chomsky 1995)
- Can Minimalism change the computational picture of syntax? Maybe, but first we need a precise specification.
- **Minimalist grammars** are such a formalization, developed by Ed Stabler. (Stabler 1997)
- They are expressive enough for syntax.

Syntax as Chemistry of Language

Minimalist grammars treat syntax like chemistry.

Chemistry	Syntax
atoms	words
electrons	features
molecules	sentences
stable	grammatical
unstable	ungrammatical

- Every word is a collection of features.
- Every feature has either positive or negative polarity.
- Features of opposite polarity annihilate each other.
- Feature annihilation drives the structure-building operations **Merge** and **Move**.

Syntax as Chemistry of Language

Minimalist grammars treat syntax like chemistry.

Chemistry	**Syntax**
atoms	words
electrons	features
molecules	sentences
stable	grammatical
unstable	ungrammatical

- Every word is a collection of features.
- Every feature has either positive or negative polarity.
- Features of opposite polarity annihilate each other.
- Feature annihilation drives the structure-building operations **Merge** and **Move**.

Merge: Example 1

Assembling [$_{DP}$ the men]

$$\frac{\text{the}}{N^{+}\ D^{-}} \quad \frac{\text{men}}{N^{-}}$$

- Features of opposite polarities annihilate
- Annihilation triggers Merge, which builds structure on top

Merge: Example 1

Assembling [$_{\text{DP}}$ the men]

$$\frac{\text{the}}{N^{+}\ D^{-}} \quad \frac{\text{men}}{N^{-}}$$

- Features of opposite polarities annihilate
- Annihilation triggers Merge, which builds structure on top

Merge: Example 1

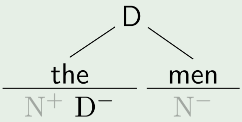
Assembling [$_{DP}$ the men]

$$\frac{\text{the}}{N^{+}\ D^{-}} \quad \frac{\text{men}}{N^{-}}$$

- Features of opposite polarities annihilate
- Annihilation triggers Merge, which builds structure on top

Merge: Example 1

Assembling [$_{DP}$ the men]



- Features of opposite polarities annihilate
- Annihilation triggers Merge, which builds structure on top

Merge: Example 1

Assembling [$_{DP}$ the men]

- Features of opposite polarities annihilate
- Annihilation triggers Merge, which builds structure on top

Merge: Example 2

Assembling [$_{VP}$ the men like which men]

the	men	like	which	men
N^+ D^-	N^-	D^+ D^+ V^-	N^+ D^-	N^-

- *the* and *men* merged as before
- same steps for *which men*
- *like* merged with *which men*
- *like* merged with *the men*

Merge: Example 2

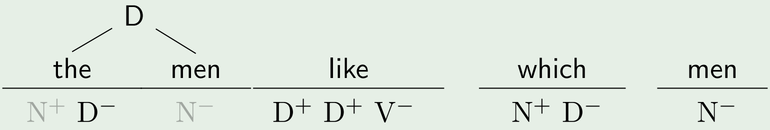
Assembling [$_{\text{VP}}$ the men like which men]

the	men	like	which	men
$N^+ \; D^-$	N^-	$D^+ \; D^+ \; V^-$	$N^+ \; D^-$	N^-

- *the* and *men* merged as before
- same steps for *which men*
- *like* merged with *which men*
- *like* merged with *the men*

Merge: Example 2

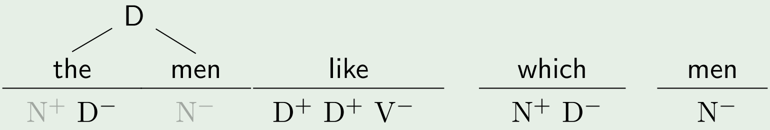
Assembling [$_{VP}$ the men like which men]



- *the* and *men* merged as before
- same steps for *which men*
- *like* merged with *which men*
- *like* merged with *the men*

Merge: Example 2

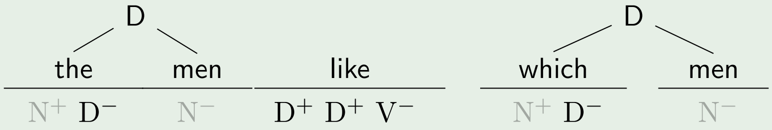
Assembling [$_{VP}$ the men like which men]



- *the* and *men* merged as before
- same steps for *which men*
- *like* merged with *which men*
- *like* merged with *the men*

Merge: Example 2

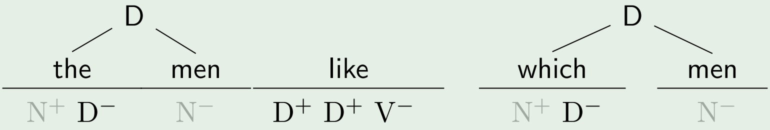
Assembling [$_{VP}$ the men like which men]



- *the* and *men* merged as before
- same steps for *which men*
- *like* merged with *which men*
- *like* merged with *the men*

Merge: Example 2

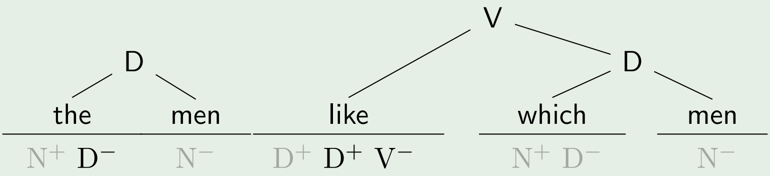
Assembling [$_{VP}$ the men like which men]



- *the* and *men* merged as before
- same steps for *which men*
- *like* merged with *which men*
- *like* merged with *the men*

Merge: Example 2

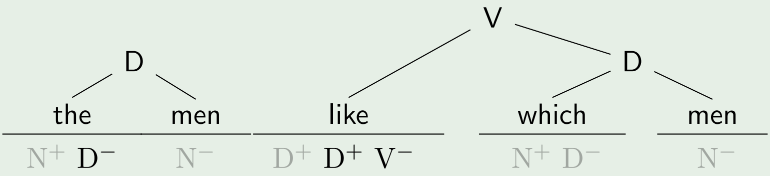
Assembling [$_{VP}$ the men like which men]



- *the* and *men* merged as before
- same steps for *which men*
- *like* merged with *which men*
- *like* merged with *the men*

Merge: Example 2

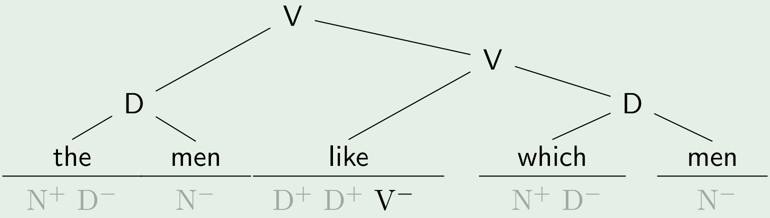
Assembling [$_{\text{VP}}$ the men like which men]



- *the* and *men* merged as before
- same steps for *which men*
- *like* merged with *which men*
- *like* merged with *the men*

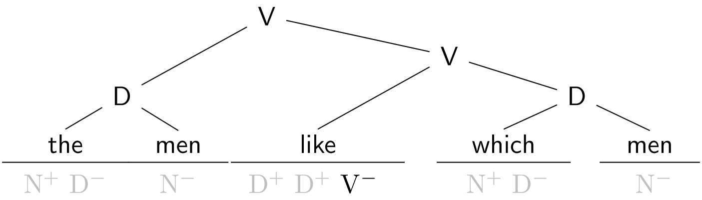
Merge: Example 2

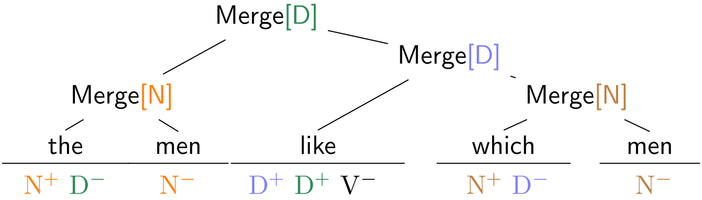
Assembling [$_{VP}$ the men like which men]



- *the* and *men* merged as before
- same steps for *which men*
- *like* merged with *which men*
- *like* merged with *the men*

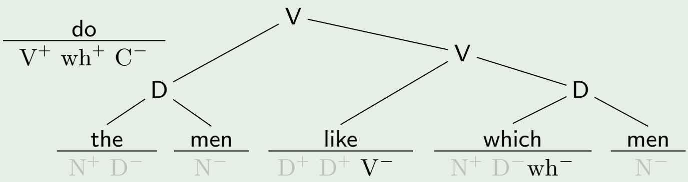
Merge: Example 2 [cont.]





Move

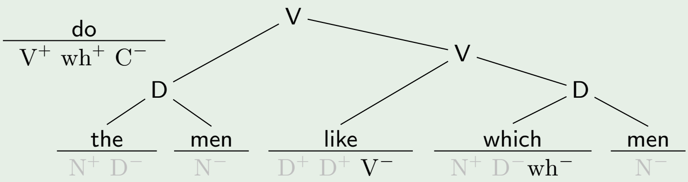
Assembling "which men do the men like?"



- Merge *do*
- Move triggered by features of opposite polarity

Move

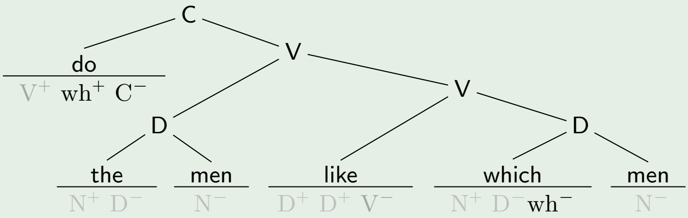
Assembling "which men do the men like?"



- Merge *do*
- Move triggered by features of opposite polarity

Move

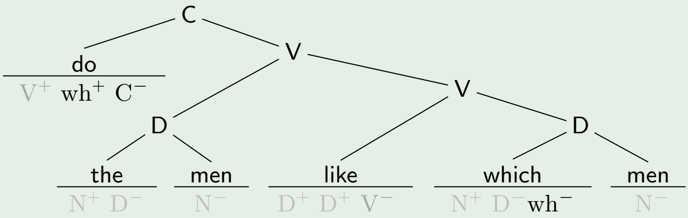
Assembling “which men do the men like?”



- Merge *do*
- Move triggered by features of opposite polarity

Move

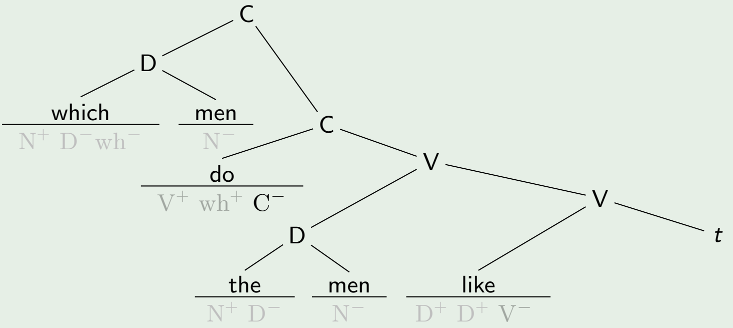
Assembling "which men do the men like?"



- Merge *do*
- Move triggered by features of opposite polarity

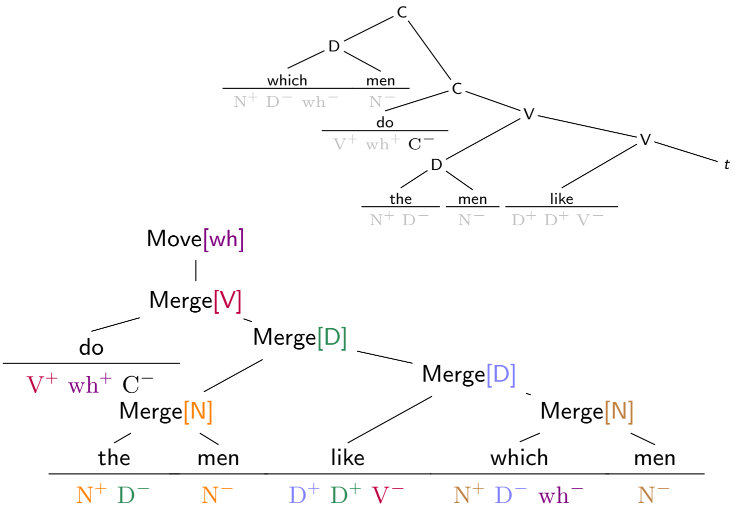
Move

Assembling "which men do the men like?"



- Merge *do*
- Move triggered by features of opposite polarity

Derivation Trees with Move

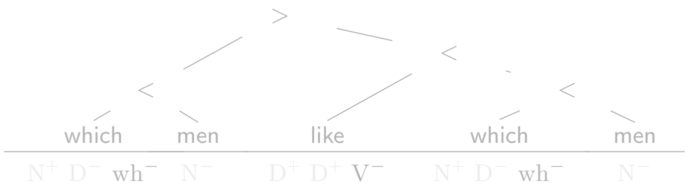


An Important Restriction on MGs

In order to ensure that MGs generate only MCS-languages, movement must be restricted.

Shortest Move Constraint (SMC)

If two lexical items in a tree both have a negative Move feature as their first currently unchecked feature, then these features must be distinct.

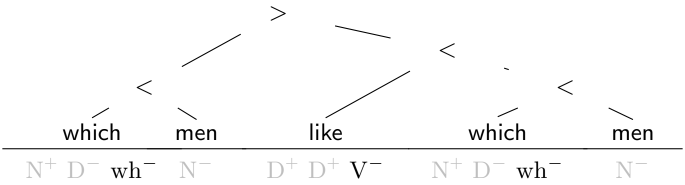


An Important Restriction on MGs

In order to ensure that MGs generate only MCS-languages, movement must be restricted.

Shortest Move Constraint (SMC)

If two lexical items in a tree both have a negative Move feature as their first currently unchecked feature, then these features must be distinct.

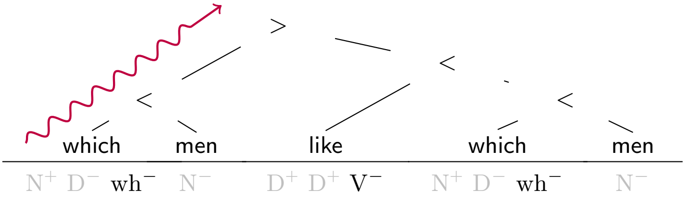


An Important Restriction on MGs

In order to ensure that MGs generate only MCS-languages, movement must be restricted.

Shortest Move Constraint (SMC)

If two lexical items in a tree both have a negative Move feature as their first currently unchecked feature, then these features must be distinct.

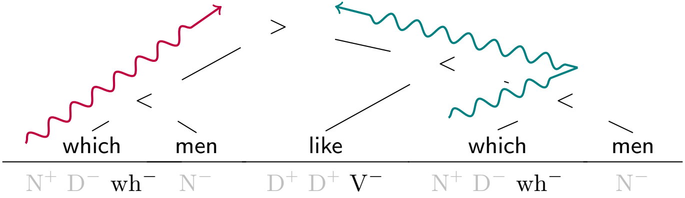


An Important Restriction on MGs

In order to ensure that MGs generate only MCS-languages, movement must be restricted.

Shortest Move Constraint (SMC)

If two lexical items in a tree both have a negative Move feature as their first currently unchecked feature, then these features must be distinct.

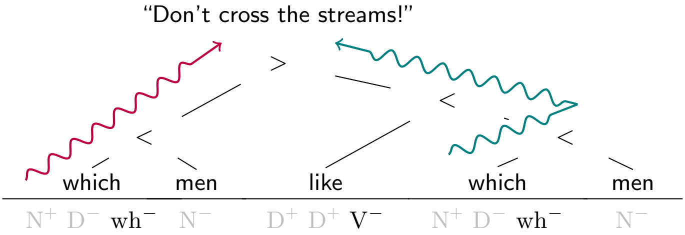


An Important Restriction on MGs

In order to ensure that MGs generate only MCS-languages, movement must be restricted.

Shortest Move Constraint (SMC)

If two lexical items in a tree both have a negative Move feature as their first currently unchecked feature, then these features must be distinct.



What's the Point?

- Sentences aren't just strings, they contain hidden structure.
- Syntacticians usually look at the tree structure that is built by the operations Merge and Move.
- **But:** the history of how such a structure is built is also a tree
  $\Rightarrow$ **phrase structure trees** and **derivation trees** as two possible views of tree-based syntax

The Complexity of MGs

- Since syntax is described by trees, we should look at tree languages instead of string languages.
- Every MG can be identified with
  - its set of phrase structure trees, or
  - its set of derivation trees
- The set of phrase structure trees is not regular. (Doner 1970; Thatcher 1967)
- But the set of derivation trees is regular. (Michaelis 2001; Kobele et al. 2007; Graf 2012)

The Big Question

Are MG derivation tree languages tier-based strictly local?

The Complexity of MGs

- Since syntax is described by trees, we should look at tree languages instead of string languages.
- Every MG can be identified with
  - its set of phrase structure trees, or
  - its set of derivation trees
- The set of phrase structure trees is not regular. (Doner 1970; Thatcher 1967)
- But the set of derivation trees is regular. (Michaelis 2001; Kobele et al. 2007; Graf 2012)

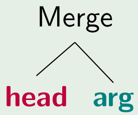
The Big Question

Are MG derivation tree languages tier-based strictly local?

Tree *n*-gram Grammars

- We need to lift *n*-grams from strings to trees.
- Instead of strings of length *n*, use subtrees of depth *n*.
- Each subtree encodes a constraint on the derivation.

Example: Bigram template for merging only matching LIs



where **head** and **arg** lack matching Merge features

Merge is Strictly Local

- An LI's Merge features are checked by its arguments.
- The distance between a head and the head of its argument is bounded by some factor k
  (which depends on how many arguments a head may take).
- Hence Merge dependencies are n-local for some fixed n.
- It follows that **Merge is tier-based strictly local**.

Constraints on Move

Suppose our MG is in **single movement normal form**.

Occurrence

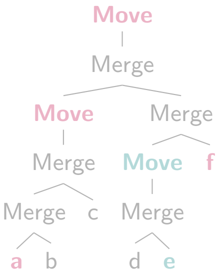
Given lexical item *l* with negative Move feature f^-, a node *m* is an **occurrence** of *l* iff *m* is the lowest node dominating *l* that can check f^-.

Movement Dependencies

- **Move**
  Every lexical item with a negative Move feature has exactly one occurrence.
- **SMC**
  Every Move node is an occurrence of exactly one lexical item.

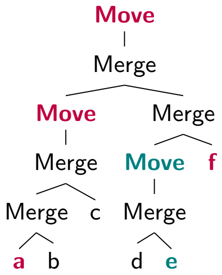
Tiers for Movement

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



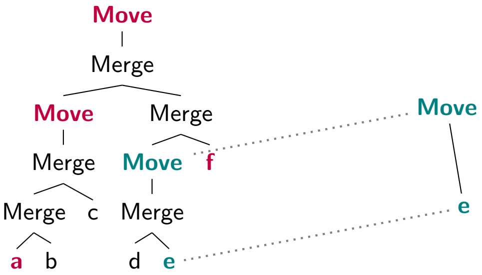
Tiers for Movement

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



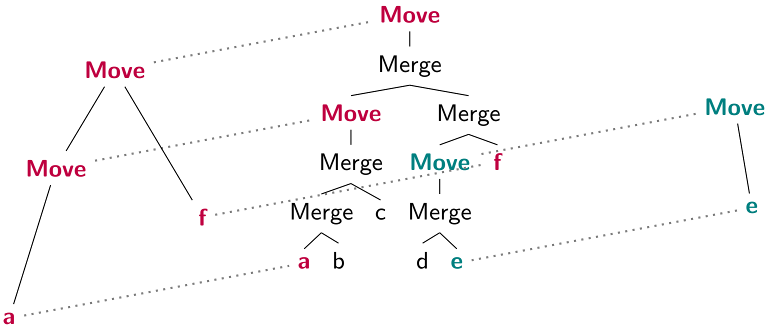
Tiers for Movement

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



Tiers for Movement

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

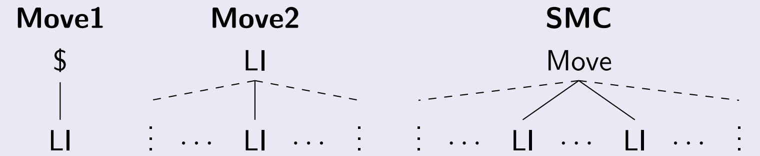


Tree Bigrams for Move

Move amounts to the following constraints over each tier:

- **Move**
  Every lexical item has a mother labeled Move.
- **SMC**
  Among the daughters of a Move node there exists exactly one that is a lexical item.

Tree Bigram Templates

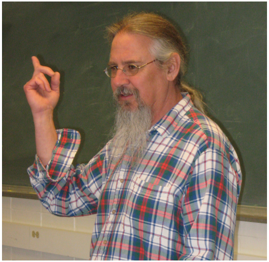


The Problem With Our Bigrams

- No limit on number of daughters per Move node in tier
  ⇒ **Move2** and **SMC** correspond to infinitely many bigrams
- But a bigram grammar must be finite!

A Hint from Multidimensional Trees

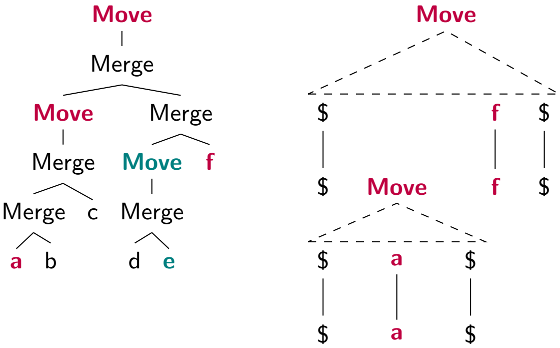
- We think of trees as nodes ordered by dominance and precedence.
- Jim Rogers (2003) formalizes trees as strings (sequences of siblings) related by dominance.
- Analogously, a tree-tier may consist of **string-tiers related by dominance**!



Checking the Example Derivation

General Verification Procedure

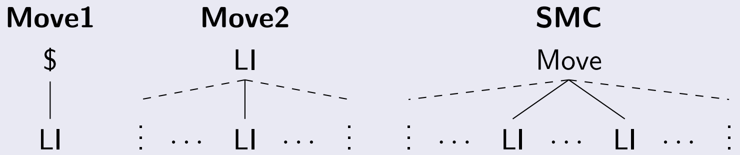
- Take derivation and project Move tiers.
- In every Move tier, project LI-tiers.
- For every node, build a bigram consisting of the node and the LI-tier of its daughter string.



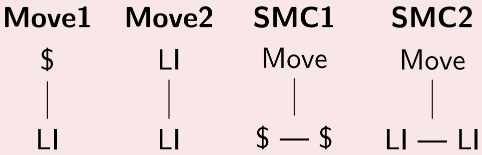
From Templates to Tree Bigrams with String Tiers

- Each string of siblings is given an LI-tier.
- The tree bigrams only reference the LI-tier.

Old Tree Bigram Templates



New Tree Bigrams with String Tiers



Conclusion

- Syntax and phonology look very different, but computationally they are very similar.
- Phonology is tier-based strictly local over strings.
- Syntax is tier-based strictly local over derivation trees.
- **Intuition**
  - Non-local dependencies are not particularly complex.
  - They are **local over some relativization domain**.

References I

Chomsky, Noam. 1956. Three models for the description of language. *IRE Transactions on Information Theory* 2:113–124.

Chomsky, Noam. 1959. On certain formal properties of grammars. *Information and Control* 2:137–167.

Chomsky, Noam. 1995. *The minimalist program*. Cambridge, Mass.: MIT Press.

Doner, John. 1970. Tree acceptors and some of their applications. *Journal of Computer and System Sciences* 4:406–451.

Graf, Thomas. 2010. *Logics of phonological reasoning*. Master's thesis, University of California, Los Angeles.

Graf, Thomas. 2012. Locality and the complexity of minimalist derivation tree languages. In *Formal Grammar 2010/2011*, ed. Philippe de Groot and Mark-Jan Nederhof, volume 7395 of *Lecture Notes in Computer Science*, 208–227. Heidelberg: Springer.

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.

Kaplan, Ronald M., and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics* 20:331–378.

References II

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

Michaelis, Jens. 2001. Transforming linear context-free rewriting systems into minimalist grammars. *Lecture Notes in Artificial Intelligence* 2099:228–244.

Rogers, James. 2003. Syntactic structures as multi-dimensional trees. *Research on Language and Computation* 1:265–305.

Shieber, Stuart M. 1985. Evidence against the context-freeness of natural language. *Linguistics and Philosophy* 8:333–345.

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

Thatcher, James W. 1967. Characterizing derivation trees for context-free grammars through a generalization of finite automata theory. *Journal of Computer and System Sciences* 1:317–322.