A Single Movement Normal Form for Minimalist Grammars

Thomas Graf Aniello De Santo Alëna Aksënova

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

FG 2016
Aug 21, 2016

Take Home Message

A mundane result...

To simplify proofs, we define a strongly equivalent normal form for MGs where every phrase **moves at most once**.

...opens many new research avenues!

- Computational parallels between syntax and phonology
- More direct connection to Dependency Grammar
- New approach to MCFL learning

Take Home Message

A mundane result...

To simplify proofs, we define a strongly equivalent normal form for MGs where every phrase **moves at most once**.

...opens many new research avenues!

- Computational parallels between syntax and phonology
- More direct connection to Dependency Grammar
- New approach to MCFL learning

Outline

Minimalist Grammars (MGs)

- Minimalist grammars (MGs) are a formalization of Chomskyan syntax (Stabler 1997, 2011)
- Succinct formalism for defining MCFGs
- Operations: Merge and **Move**
- Grammar is just a finite list of feature-annotated lexical items (LIs)

Chemistry	**Syntax**
atoms	words
electrons	features
molecules	sentences

Merge

Merge combines subtrees to encode **head-argument dependencies**.

category feature $\mathrm{N}^{-}, \mathrm{V}^{-}, \ldots$

selector feature $\mathrm{N}^{+}, \mathrm{V}^{+}, \ldots$

the	men	like	which	men
$\mathrm{N}^{+}\ \mathrm{D}^{-}$	N^{-}	$\mathrm{D}^{+}\ \mathrm{D}^{+}\ \mathrm{V}^{-}$	$\mathrm{N}^{+}\ \mathrm{D}^{-}$	N^{-}

- *the* and *men* have matching features, triggering Merge
- same steps for *which men*
- *like* merged with *which men*
- *like* merged with *the men*

Merge

Merge combines subtrees to encode **head-argument dependencies**.

category feature N^{-}, V^{-}, ...

selector feature N^{+}, V^{+}, ...

the	men	like	which	men
$\mathrm{N}^{+}\ \mathrm{D}^{-}$	N^{-}	$\mathrm{D}^{+}\ \mathrm{D}^{+}\ \mathrm{V}^{-}$	$\mathrm{N}^{+}\ \mathrm{D}^{-}$	N^{-}

- *the* and *men* have matching features, triggering Merge
- same steps for *which men*
- *like* merged with *which men*
- *like* merged with *the men*

Merge

Merge combines subtrees to encode **head-argument dependencies**.

category feature N^-, V^-, ...

selector feature N^+, V^+, ...

$<$

the	men	like	which	men
$\mathrm{N}^+\ \mathrm{D}^-$	N^-	$\mathrm{D}^+\ \mathrm{D}^+\ \mathrm{V}^-$	$\mathrm{N}^+\ \mathrm{D}^-$	N^-

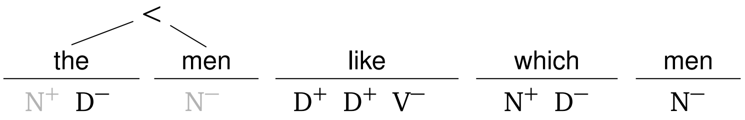
- *the* and *men* have matching features, triggering Merge
- same steps for *which men*
- *like* merged with *which men*
- *like* merged with *the men*

Merge

Merge combines subtrees to encode **head-argument dependencies**.

category feature $\mathrm{N}^{-}, \mathrm{V}^{-}, \ldots$

selector feature $\mathrm{N}^{+}, \mathrm{V}^{+}, \ldots$



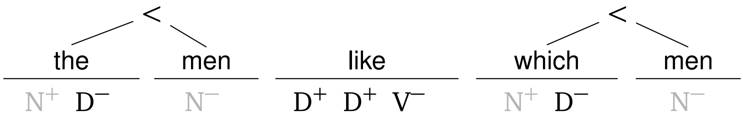
- *the* and *men* have matching features, triggering Merge
- same steps for *which men*
- *like* merged with *which men*
- *like* merged with *the men*

Merge

Merge combines subtrees to encode **head-argument dependencies**.

category feature N^-, V^-, ...

selector feature N^+, V^+, ...



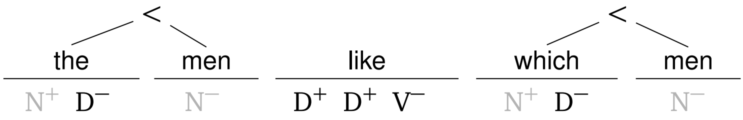
- *the* and *men* have matching features, triggering Merge
- same steps for *which men*
- *like* merged with *which men*
- *like* merged with *the men*

Merge

Merge combines subtrees to encode **head-argument dependencies**.

category feature N^-, V^-, ...

selector feature N^+, V^+, ...



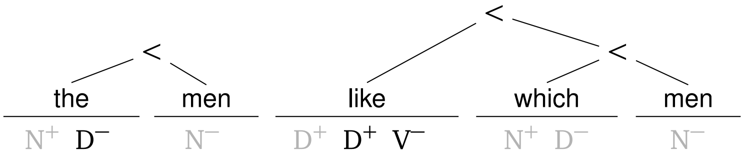
- *the* and *men* have matching features, triggering Merge
- same steps for *which men*
- *like* merged with *which men*
- *like* merged with *the men*

Merge

Merge combines subtrees to encode **head-argument dependencies**.

category feature N^- , V^- , ...

selector feature N^+ , V^+ , ...



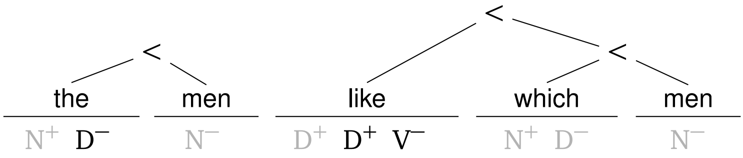
- *the* and *men* have matching features, triggering Merge
- same steps for *which men*
- *like* merged with *which men*
- *like* merged with *the men*

Merge

Merge combines subtrees to encode **head-argument dependencies**.

category feature N^- , V^- , . . .

selector feature N^+ , V^+ , . . .



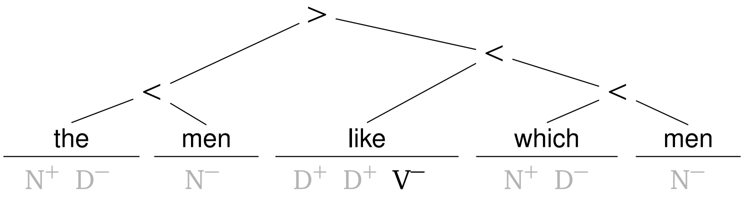
- *the* and *men* have matching features, triggering Merge
- same steps for *which men*
- *like* merged with *which men*
- *like* merged with *the men*

Merge

Merge combines subtrees to encode **head-argument dependencies**.

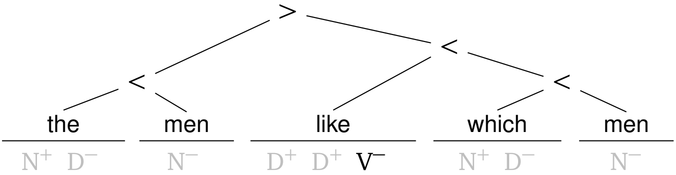
category feature N^-, V^-, ...

selector feature N^+, V^+, ...

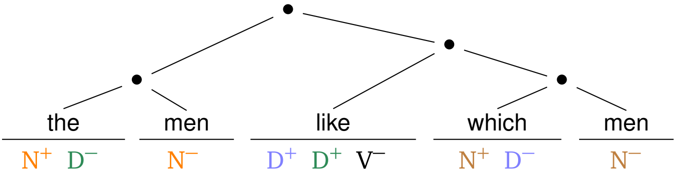


- *the* and *men* have matching features, triggering Merge
- same steps for *which men*
- *like* merged with *which men*
- *like* merged with *the men*

Merge in Derivation Trees



Derived Tree



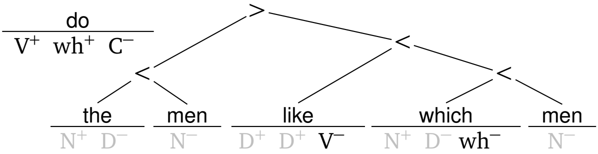
Derivation Tree

Move

Move displaces subtrees to derive the correct **linear order**.

licensee feature wh^- , top^- , ...

licensor feature wh^+ , top^+ , ...



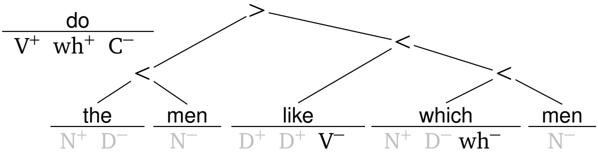
- Merge *do*
- Move triggered by features of opposite polarity

Move

Move displaces subtrees to derive the correct **linear order**.

licensee feature wh^- , top^- , . . .

licensor feature wh^+ , top^+ , . . .



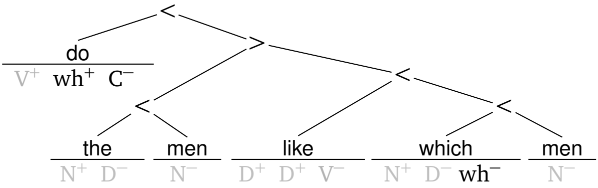
- Merge *do*
- Move triggered by features of opposite polarity

Move

Move displaces subtrees to derive the correct **linear order**.

licensee feature wh^- , top^- , ...

licensor feature wh^+ , top^+ , ...



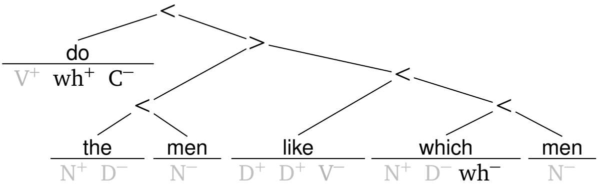
- Merge *do*
- Move triggered by features of opposite polarity

Move

Move displaces subtrees to derive the correct **linear order**.

licensee feature wh^-, top^-, ...

licensor feature wh^+, top^+, ...



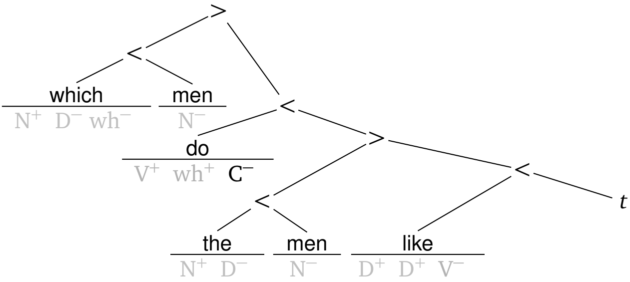
- Merge *do*
- Move triggered by features of opposite polarity

Move

Move displaces subtrees to derive the correct **linear order**.

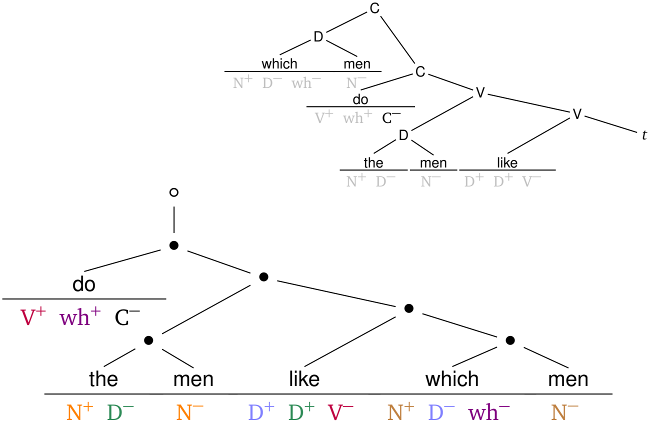
licensee feature wh^- , top^- , ...

licensor feature wh^+ , top^+ , ...

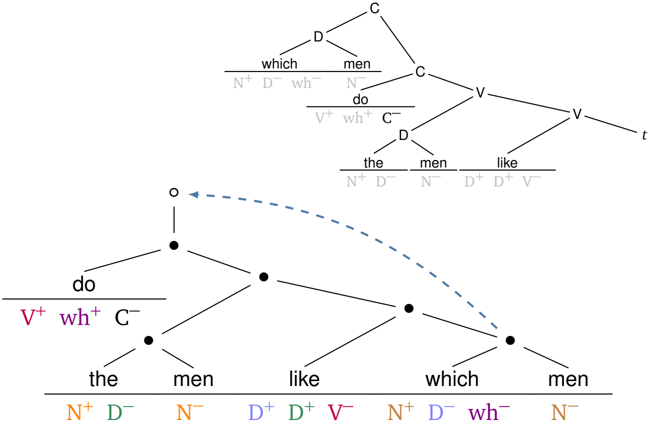


- Merge *do*
- Move triggered by features of opposite polarity

Move in Derivation Trees

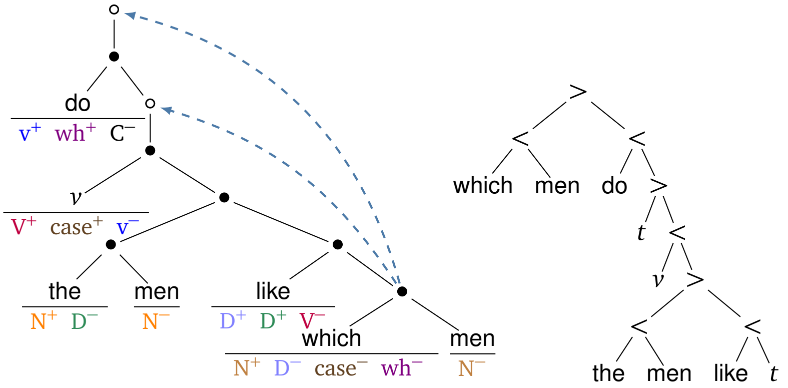


Move in Derivation Trees



Intermediate Movement

Intermediate Movement is possible, but does not affect string order.

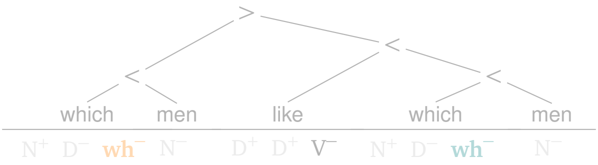


An Important Restriction on MGs

In order to ensure that MGs generate only MCFLs,
movement must be restricted.

Shortest Move Constraint (SMC)

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

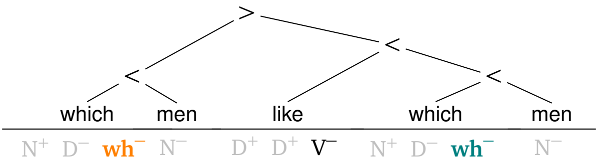


An Important Restriction on MGs

In order to ensure that MGs generate only MCFLs, movement must be restricted.

Shortest Move Constraint (SMC)

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

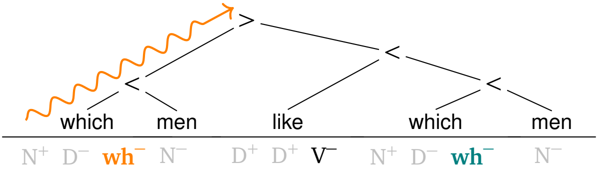


An Important Restriction on MGs

In order to ensure that MGs generate only MCFLs,
movement must be restricted.

Shortest Move Constraint (SMC)

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

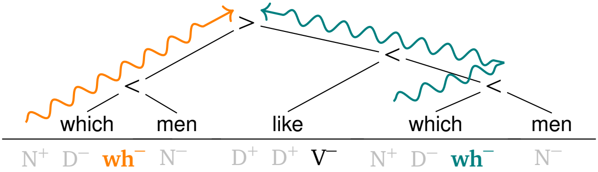


An Important Restriction on MGs

In order to ensure that MGs generate only MCFLs, movement must be restricted.

Shortest Move Constraint (SMC)

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

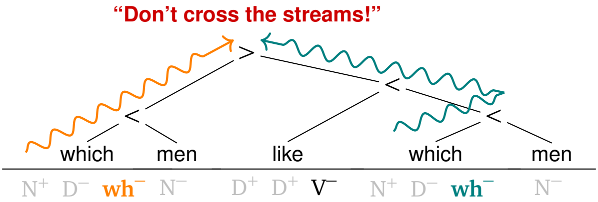


An Important Restriction on MGs

In order to ensure that MGs generate only MCFLs, movement must be restricted.

Shortest Move Constraint (SMC)

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



MGs as Minimalist Derivation Tree Languages

- Phrase structure trees are redundant.
- Every MG can be equated with its well-formed derivations, its **Minimalist Derivation Tree Language (MDTL)**:

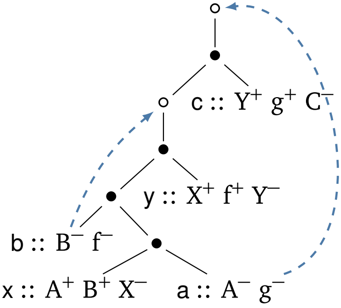
Merge	Merge features must be checked.
Move	Move features must be checked.
SMC	SMC must not be violated.
Lex	The set of LIs must be finite.
Max	MDTL must contain every well-formed derivation over the lexicon.

- Every MDTL is a regular tree language. (Michaelis 2001; Kobele et al. 2007; Graf 2012)

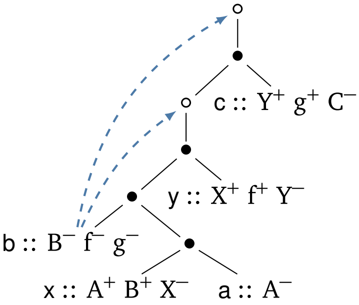
Definition

Definition (SMNF)

An MG G is in **single movement normal form (SMNF)** iff every LI of G has at most one licensee feature.



SMNF



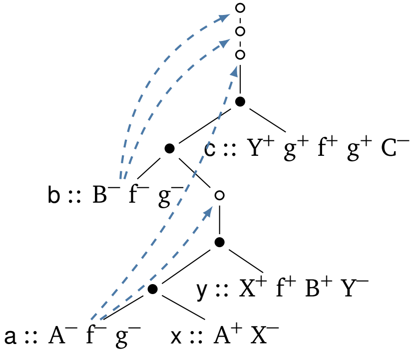
not SMNF

A Failed Attempt

Feature Atomization

If an LI's string of licensee features is $\delta := f_1^- \cdots f_n^-$,
then replace δ by $[f_1 \cdots f_n]^-$.

This causes **SMC violations**:

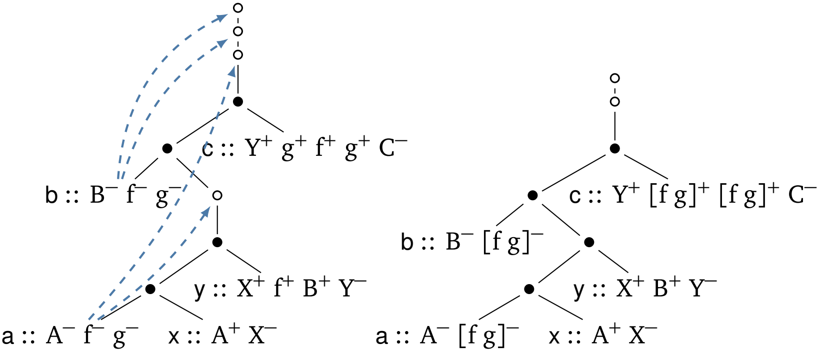


A Failed Attempt

Feature Atomization

If an LI's string of licensee features is $\delta := f_1^- \cdots f_n^-$,
then replace δ by $[f_1 \cdots f_n]^-$.

This causes **SMC violations**:

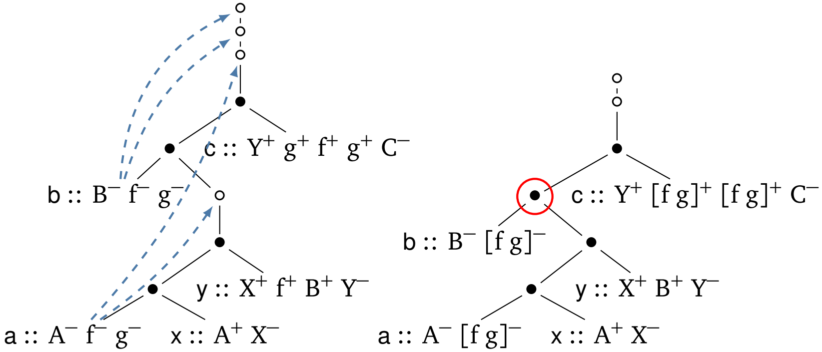


A Failed Attempt

Feature Atomization

If an LI's string of licensee features is $\delta := f_1^- \cdots f_n^-$,
then replace δ by $[f_1 \cdots f_n]^-$.

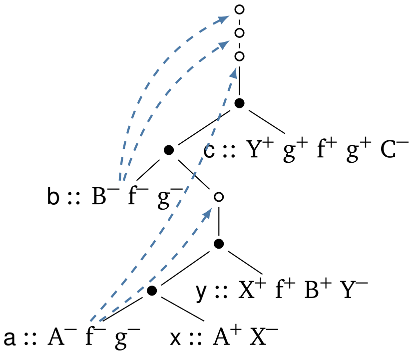
This causes **SMC violations**:



Success With Subscripts

Feature Subscripting

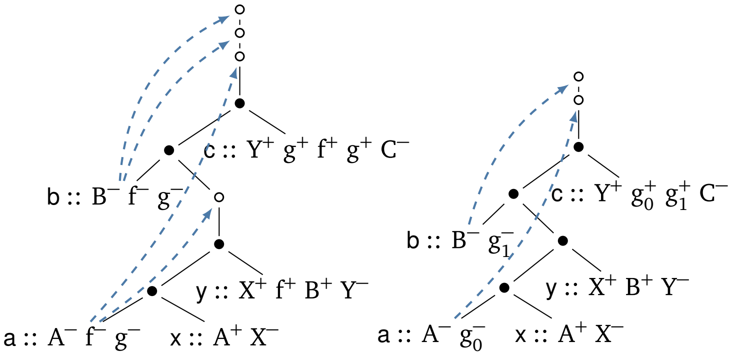
- For every LI l, only keep its last licensee feature.
- Add subscripts to licensee features to avoid SMC violations.



Success With Subscripts

Feature Subscripting

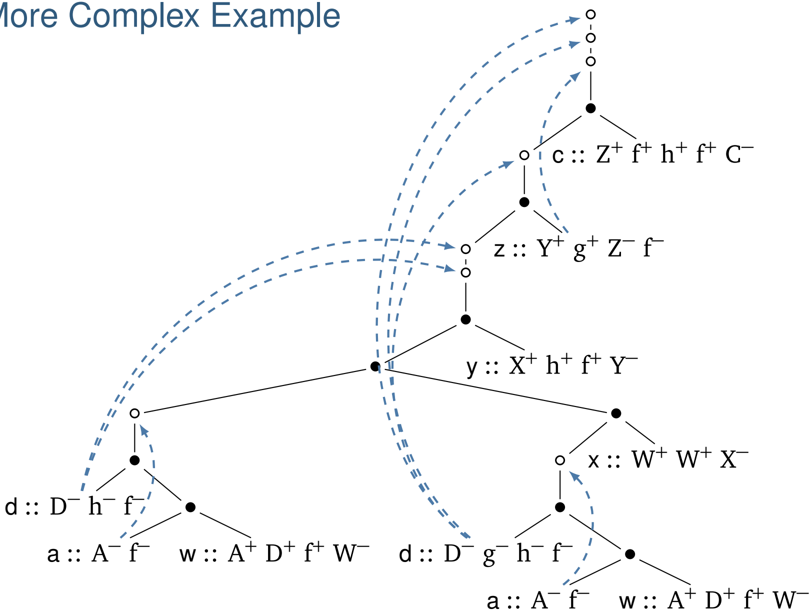
- For every LI l, only keep its last licensee feature.
- Add subscripts to licensee features to avoid SMC violations.



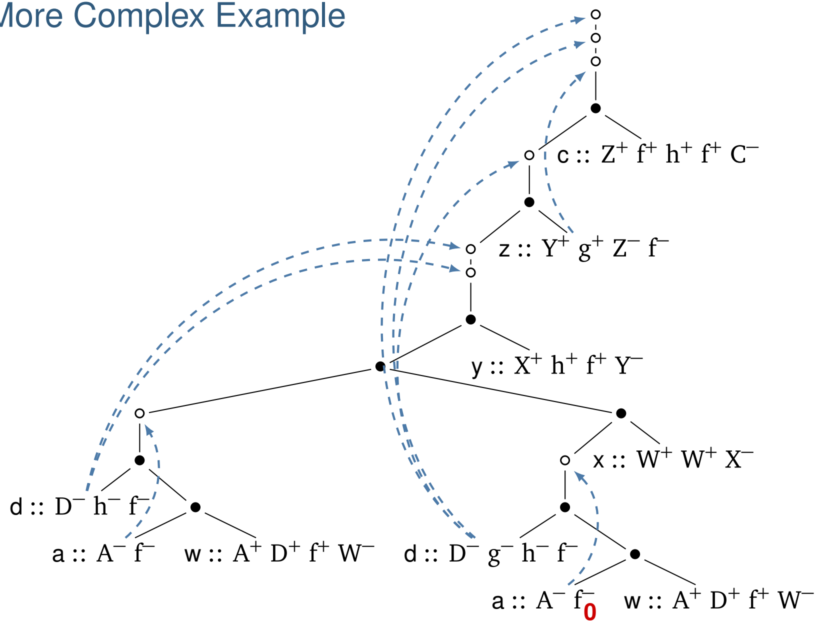
Technical Details of Procedure

- For each LI l, only keep its last licensee feature f^{-} .
- Subscript f^{-} with an index j.
- Index j must be **minimal**:
  - Assume that l belongs to Move node m.
  - For every $0 \leq i < j$, there is a LI l' that ends in $\mathrm{f}_{\mathrm{i}}^{-}$ and belongs to move node m' such that m dominates m' and m' dominates l.
- Add index j to the corresponding licensee feature f^{+} that checked f^{-} in the original derivation.
- Remove all licensor features without subscripts.

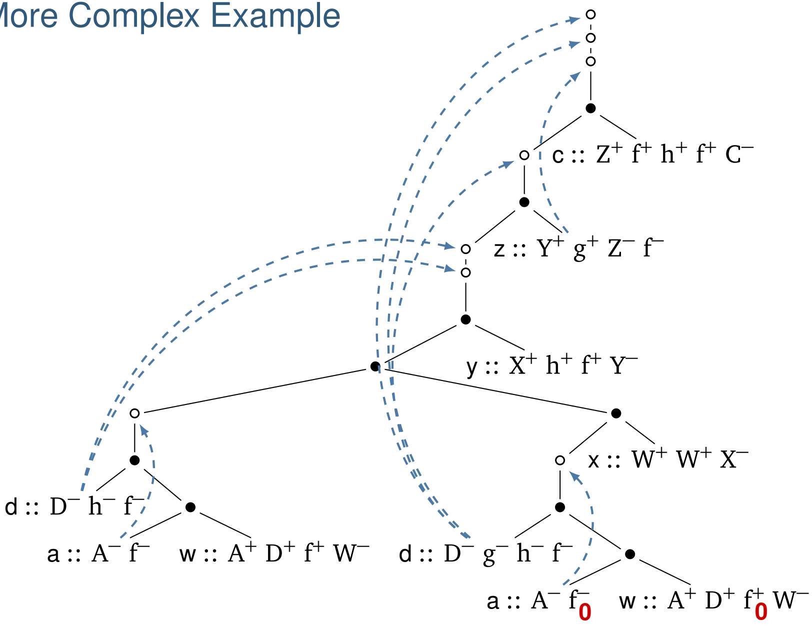
A More Complex Example



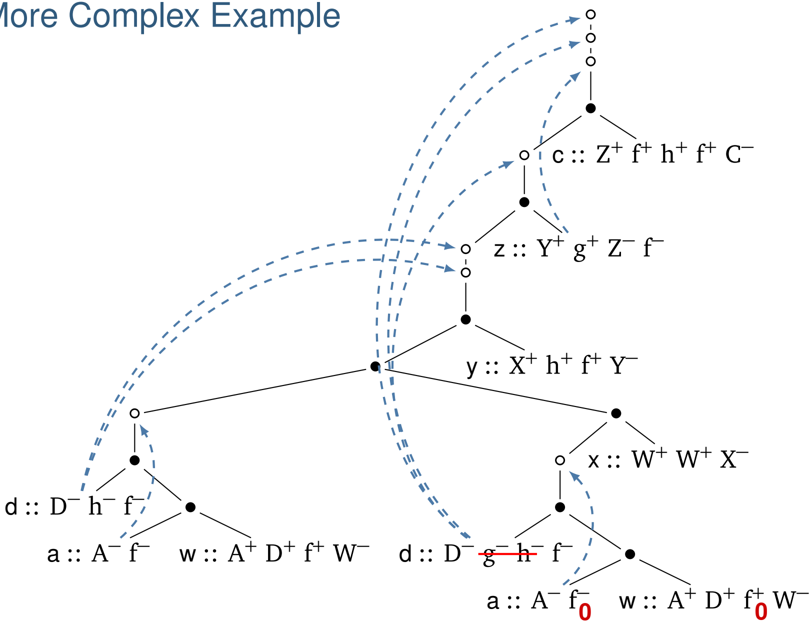
A More Complex Example



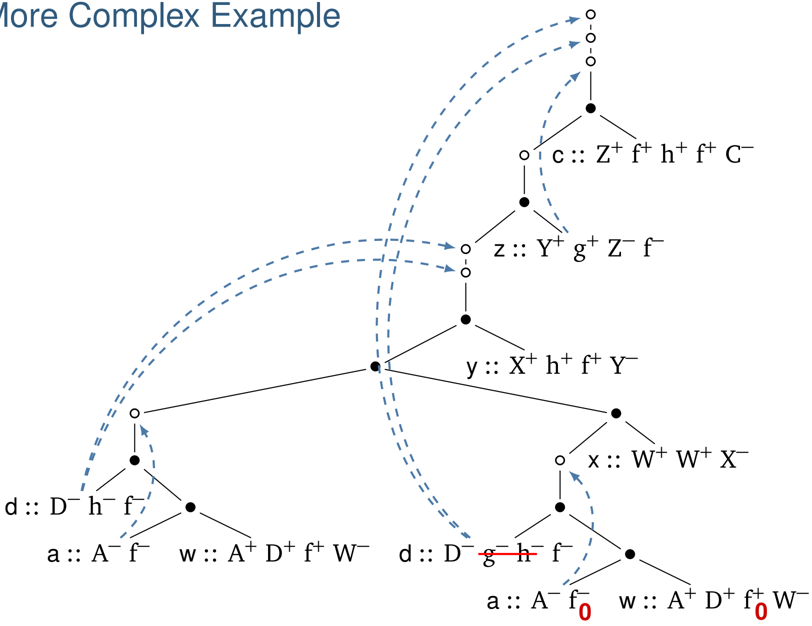
A More Complex Example



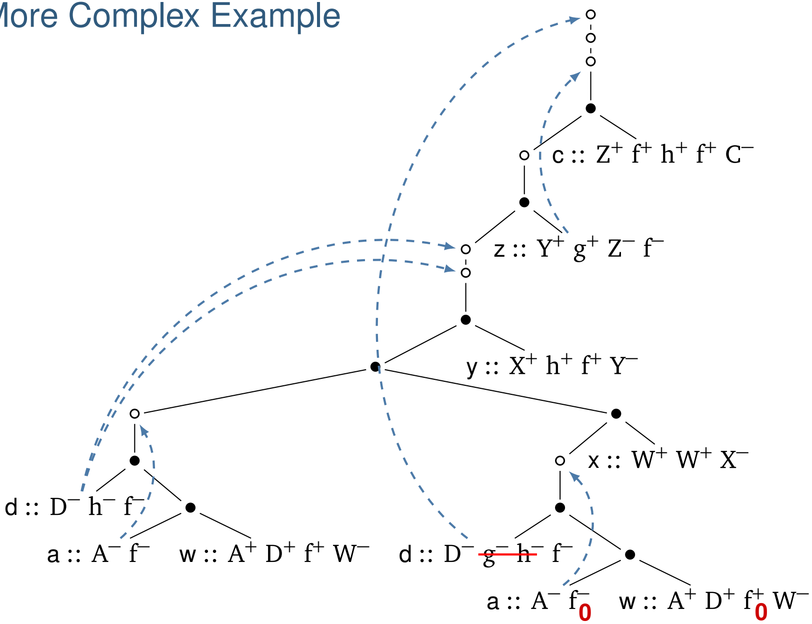
A More Complex Example



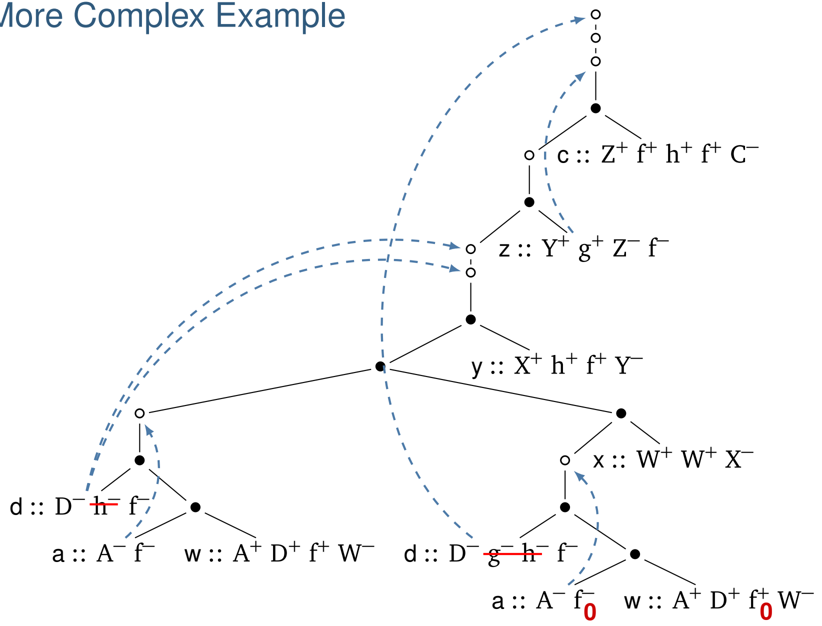
A More Complex Example



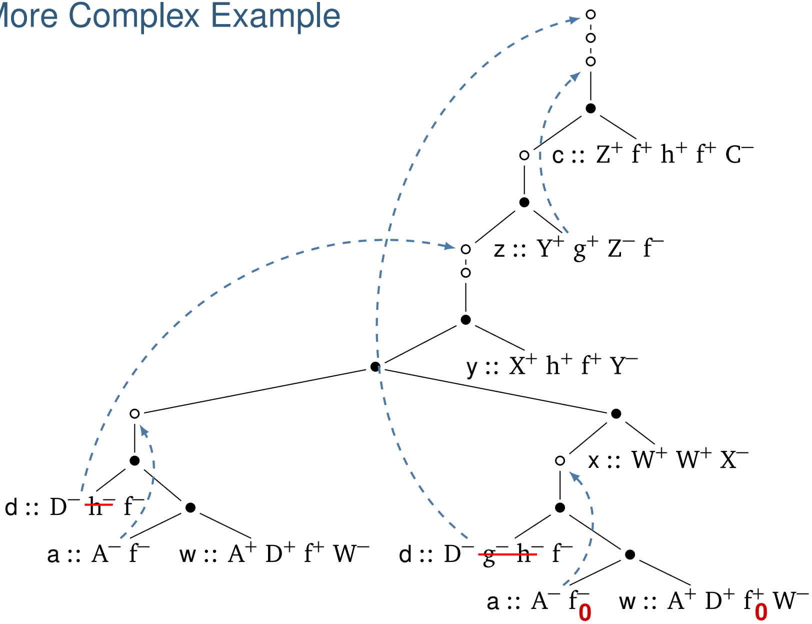
A More Complex Example



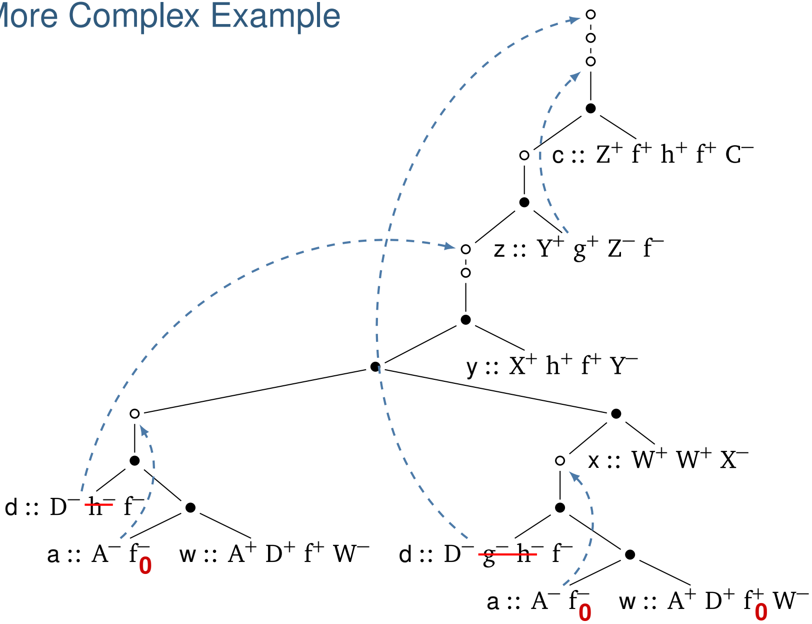
A More Complex Example



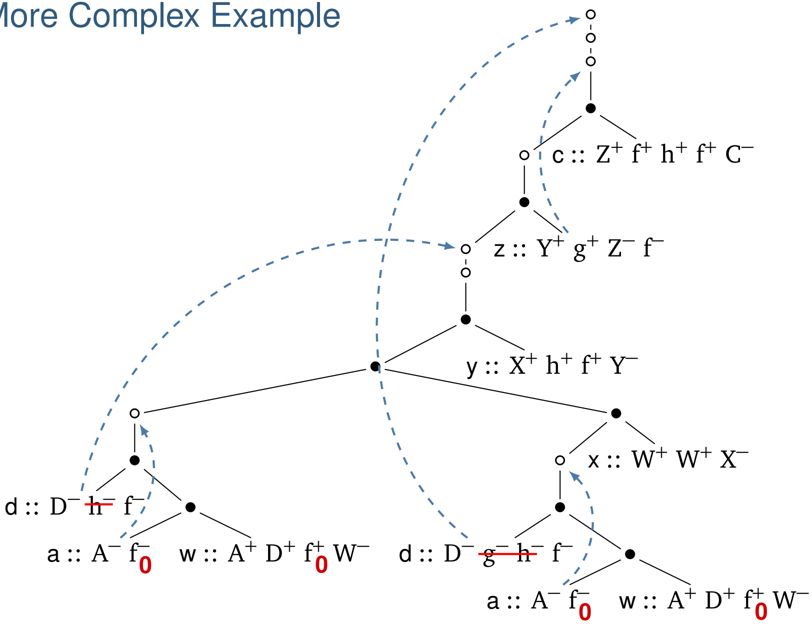
A More Complex Example



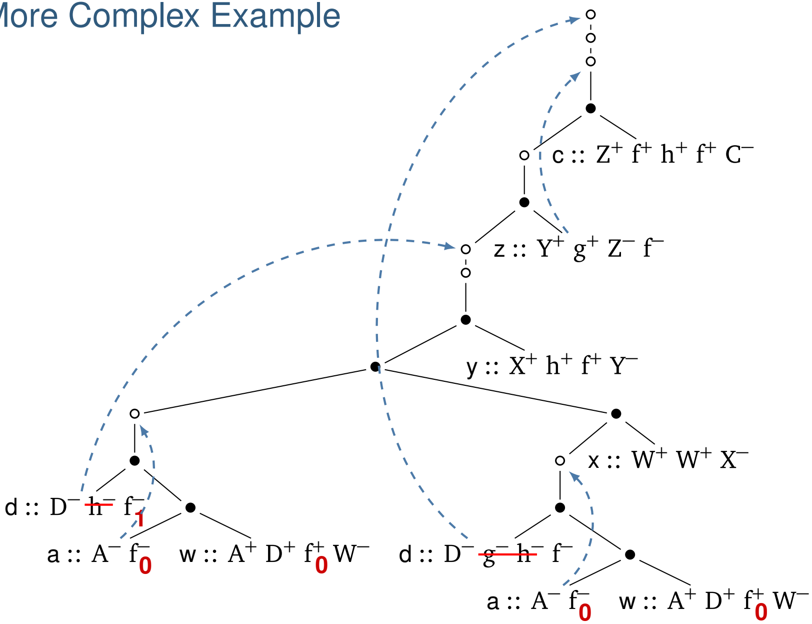
A More Complex Example



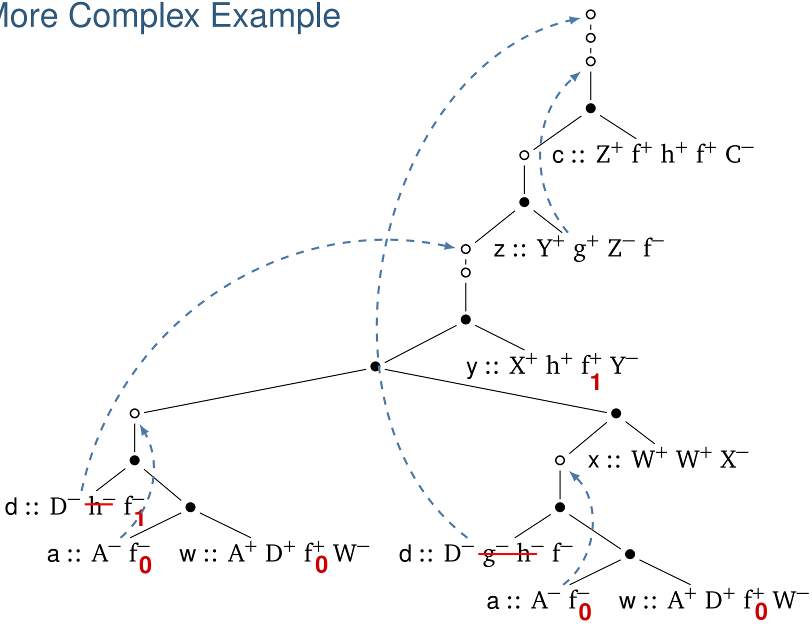
A More Complex Example



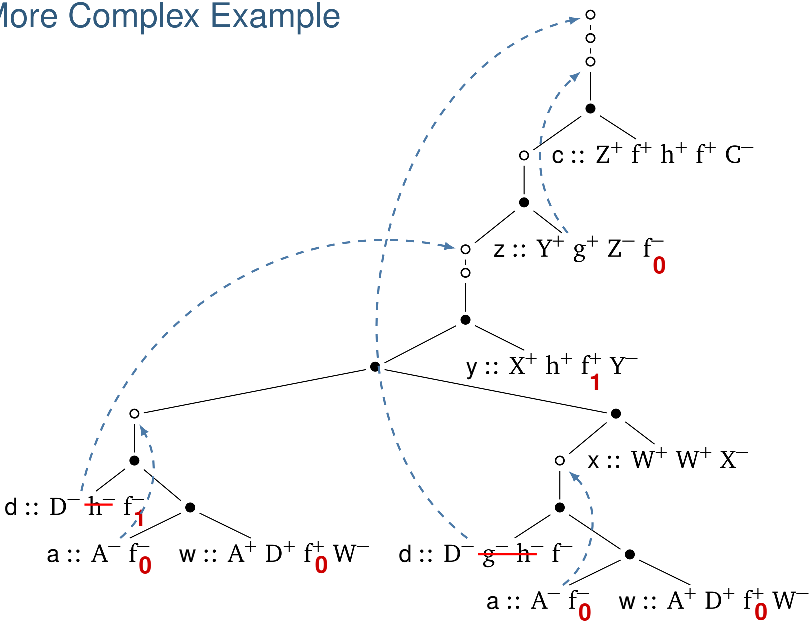
A More Complex Example



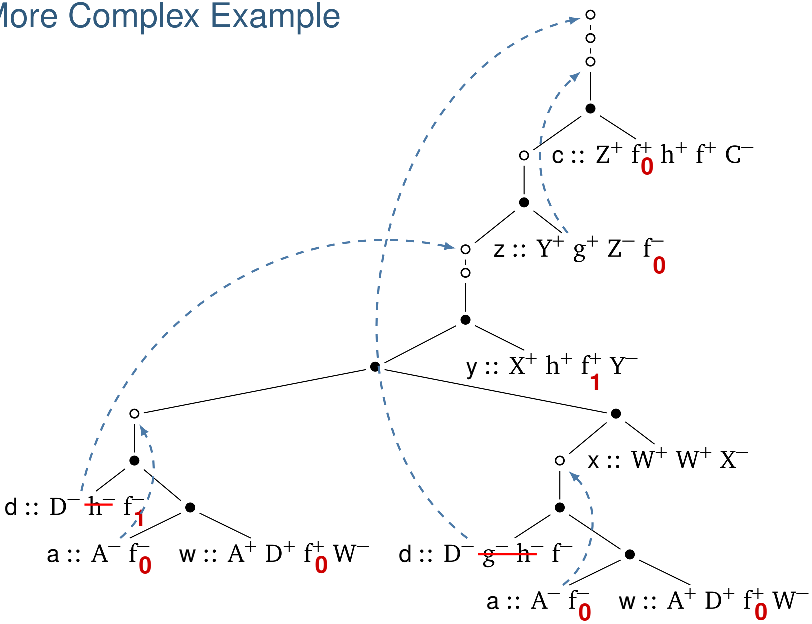
A More Complex Example



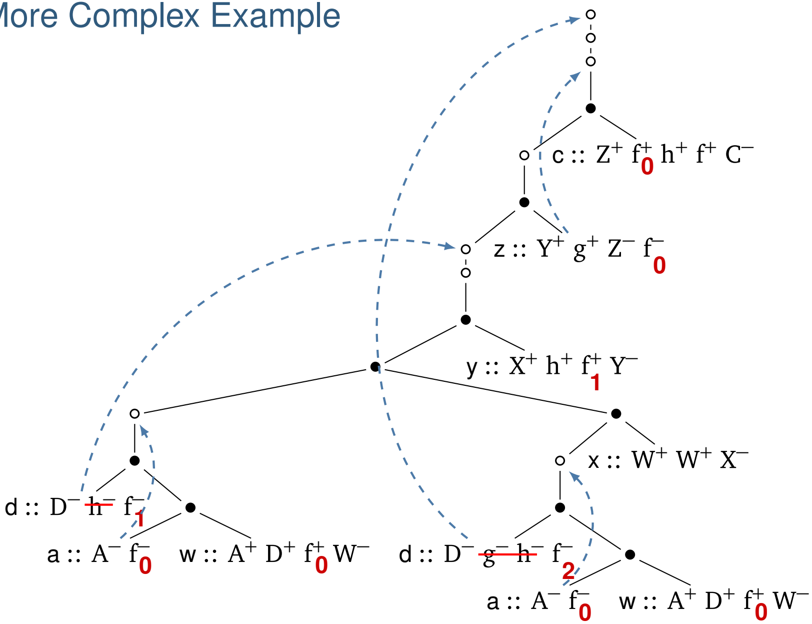
A More Complex Example



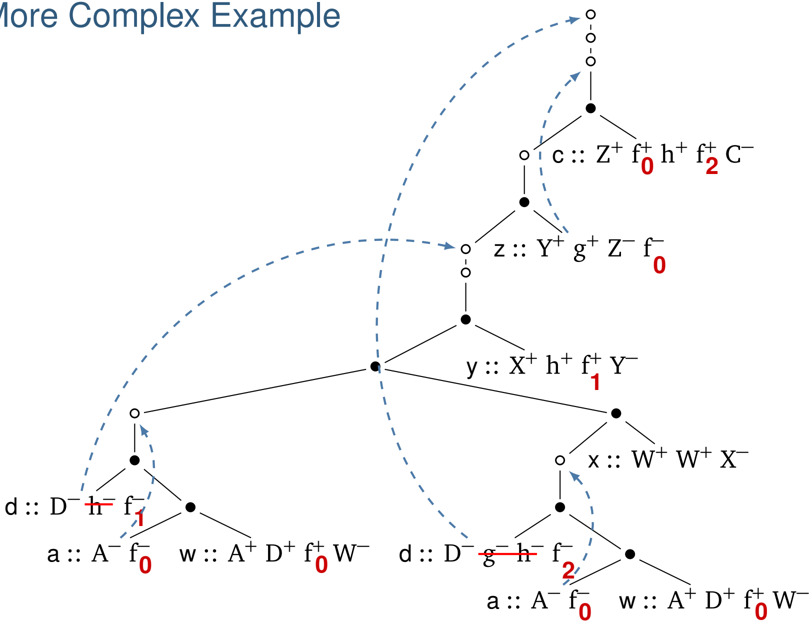
A More Complex Example



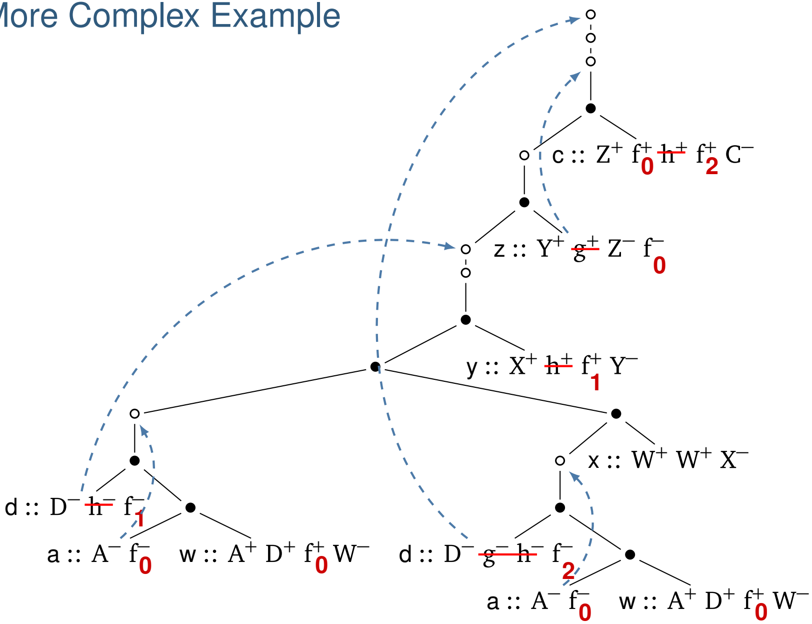
A More Complex Example



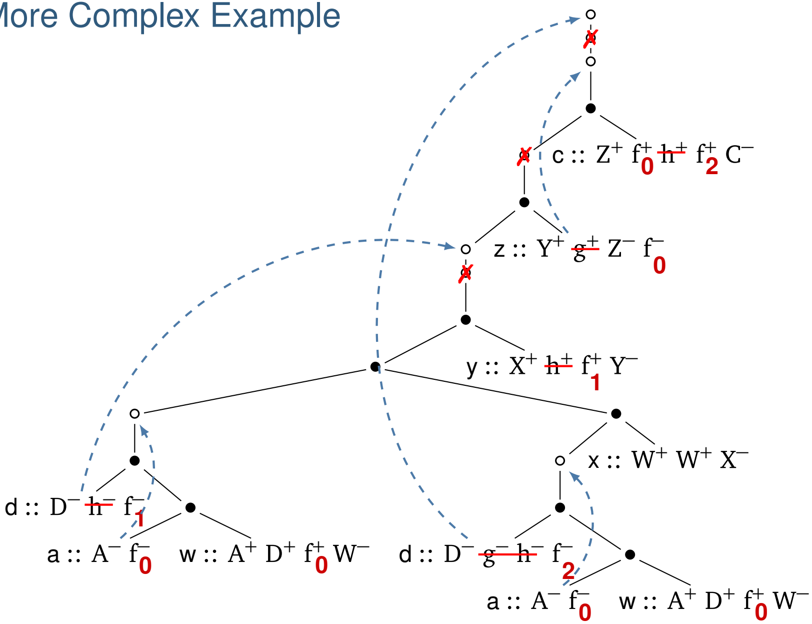
A More Complex Example



A More Complex Example



A More Complex Example



Procedure is Correct for Each Derivation

Lemma

The translation produces well-formed derivations in SMNF.

- **Merge:** unaffected by translation
- **Move:** still one-to-one matching between licensee features and licensor features in every derivation

Lemma

The translation preserves the phrase structure tree for each derivation (modulo intermediate landing sites).

- **Merge:** unaffected by translation
- **Move:** Every LI still belongs to the same final move node $\Rightarrow$ every phrase still moves to the same position as before

Procedure is Correct for Each Derivation

Lemma

The translation produces well-formed derivations in SMNF.

- **Merge:** unaffected by translation
- **Move:** still one-to-one matching between licensee features and licensor features in every derivation

Lemma

The translation preserves the phrase structure tree for each derivation (modulo intermediate landing sites).

- **Merge:** unaffected by translation
- **Move:** Every LI still belongs to the same final move node
  $\Rightarrow$ every phrase still moves to the same position as before

Procedure Yields an MG

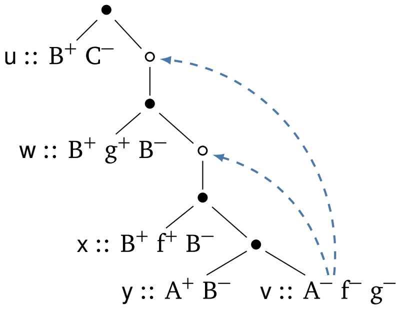
Lemma

The range of the translation procedure is almost an MDTL.

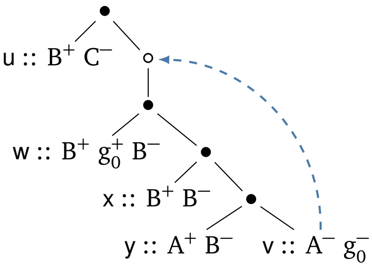
- **Set of Well-Formed Derivations:** follows from previous results
- **Finite Lexicon:**
  - SMC puts upper bound k on how many distinct subscripts are needed for each grammar
  - Consequently, each LI is refined into at most k variants.
  - Lexical blow-up finitely bounded by k
- **Regular Set of Derivation Trees:**
  - MGs have regular derivation tree languages
  - Translation carried out by linear tree transducer, which preserves regularity

Why “Almost”?

The subscripted LIs may allow for **completely new derivations**:



$L := \{uvwxy\}$



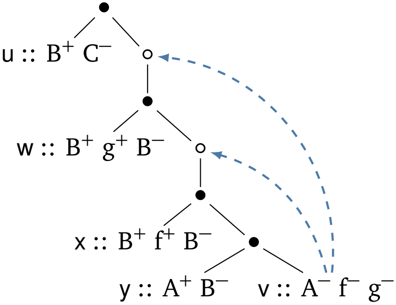
$L := uvwx^*y$

Maximality Failure

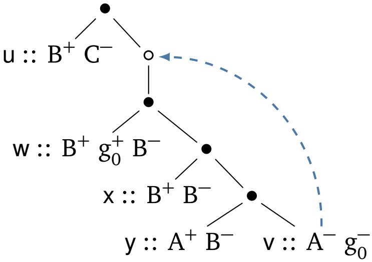
The range of the translation fails MDTLs’ **maximality requirement**.

Why “Almost”?

The subscripted LIs may allow for **completely new derivations**:



$L := \{uvwxy\}$



$L := uvwx^*y$

Maximality Failure

The range of the translation fails MDTLs’ **maximality requirement**.

Intersection to the Rescue

- For every MDTL L and regular tree language R, one can convert $L \cap R$ into an MDTL. (Graf 2011; Kobele 2011)
- Let L be the MDTL of the MG consisting of all the LIs produced by the SMNF translation.
- Let R be the range of the SMNF translation.
- Then $L \cap R$ yields the desired, strongly equivalent MDTL.

Theorem (SMNF for MGs)

For every MG there is a strongly equivalent one in SMNF.

Lexical Blow-Up

- SMNF translation induces **linear lexical blow-up**
- Effect varies a lot depending on movement configurations:

lower bound	linear size reduction(!), 1:1 for non-redundant grammars
upper bound	large linear blow-up

$$\sum_{\mathbf{l} \in \mathbf{Lex}} \mu^{\gamma(\mathbf{l}) + \delta(\mathbf{l})}$$

μ ... maximum number of required indices

$\gamma(\mathbf{l})$ ... number of licensor features of LI l in original grammar

$\delta(\mathbf{l})$ ... 1 if l has licensee features, 0 otherwise

Lexical Blow-Up [cont.]

Linguist: Support for Multiple Movement

Grammars where phrases move in several steps are more succinct and thus to be preferred.

Alternative: A New Empirical Puzzle

Are the movement configurations we find in natural language exactly those that **induce little lexical blow-up**?

$\Rightarrow$ new way of pruning MG overgeneration

Lexical Blow-Up [cont.]

Linguist: Support for Multiple Movement

Grammars where phrases move in several steps are more succinct and thus to be preferred.

Alternative: A New Empirical Puzzle

Are the movement configurations we find in natural language exactly those that **induce little lexical blow-up**?

⇒ new way of pruning MG overgeneration

Parallels Between Syntax and Phonology

- Phonology is subregular:
  tier-based strictly local
  (Heinz 2015)
- MDTLs are also subregular.
- But **only SMNF MDTLs** are also
  tier-based strictly local.
  (Graf and Heinz 2016)

Computational Parallelism Hypothesis

Syntax and phonology have comparable subregular complexity over strings and derivation trees, respectively.

- sharing of theorems, proof techniques, and NLP tools
- new learning algorithms

Parallels Between Syntax and Phonology

- Phonology is subregular:
  tier-based strictly local
  (Heinz 2015)
- MDTLs are also subregular.
- But **only SMNF MDTLs** are also tier-based strictly local.
  (Graf and Heinz 2016)

Computational Parallelism Hypothesis

Syntax and phonology have comparable subregular complexity over strings and derivation trees, respectively.

- sharing of theorems, proof techniques, and NLP tools
- new learning algorithms

Connection to Dependency Grammar

- MGs are closely connected to Dependency Grammar.
  (Boston et al. 2010)
- If one removes Move nodes from MG derivations, they are basically **dependency graphs**.
- Dependency graphs indicate linear order directly instead of Move.
- In SMNF MG, every Move nodes has visible effect on linear order
  $\Rightarrow$ easier to deduce movement from linear order

Towards a New Learning Algorithm for MGs/MCFLs

Input	text of dependency graphs with linear string order
Output	SMNF MG

Connection to Dependency Grammar

- MGs are closely connected to Dependency Grammar. (Boston et al. 2010)
- If one removes Move nodes from MG derivations, they are basically **dependency graphs**.
- Dependency graphs indicate linear order directly instead of Move.
- In SMNF MG, every Move nodes has visible effect on linear order $\Rightarrow$ easier to deduce movement from linear order

Towards a New Learning Algorithm for MGs/MCFLs

Input	text of dependency graphs with linear string order
Output	SMNF MG

Taking Stock

- MGs are all about two structure-building operations: **Merge** and **Move**.
- Intermediate movement complicates formalism
- SMNF removes it from formalism without affecting strong generative capacity
- New research opportunities:
  - exact interaction of movement and lexical blow-up
  - characterization of natural language movement in terms of blow-up bounds
  - parallels between syntax and phonology
  - connection to Dependency Grammar
  - new learning algorithm for MGs

References I

Boston, Marisa Ferrara, John T. Hale, and Marco Kuhlmann. 2010. Dependency structures derived from Minimalist grammars. In *Mol 10/11*, ed. Christian Ebert, Gerhard Jäger, and Jens Michaelis, volume 6149 of *Lecture Notes in Computer Science*, 1–12. Berlin: Springer.

Graf, Thomas. 2011. Closure properties of Minimalist derivation tree languages. In *LACL 2011*, ed. Sylvain Pogodalla and Jean-Philippe Prost, volume 6736 of *Lecture Notes in Artificial Intelligence*, 96–111. Heidelberg: Springer.

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

Graf, Thomas, and Jeffrey Heinz. 2016. Tier-based strict locality in phonology and syntax. Ms., Stony Brook University and University of Delaware.

Heinz, Jeffrey. 2015. The computational nature of phonological generalizations. URL `http://www.socsci.uci.edu/~lpearl/colareadinggroup/readings/Heinz2015BC_Typology.pdf`, ms., University of Delaware.

Kobele, Gregory M. 2011. Minimalist tree languages are closed under intersection with recognizable tree languages. In *LACL 2011*, ed. Sylvain Pogodalla and Jean-Philippe Prost, volume 6736 of *Lecture Notes in Artificial Intelligence*, 129–144.

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.

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

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