Locality and the Complexity of Minimalist Derivation Tree Languages

Thomas Graf
tgraf@ucla.edu
tgraf.bol.ucla.edu

University of California, Los Angeles

FG 2011
August 6, 2011

This Talk in a Nutshell

- **What?**
  - investigate Minimalist grammars (MGs) with respect to the structural complexity of their derivation trees
  - measure the impact of syntactic locality conditions
- **Why?**
  - *Modularity results* (Kolb et al. 2003; Kobele et al. 2007): non-CF grammar formalism = regular derivational calculus + tree yield function
  - *Short term question*: What are the properties of the MG derivation calculus? What is the role of locality in syntax?
  - *Long term question*: How do different formalisms distribute the workload over these two components? In particular, how do MGs compare to TAG?

Outline

MGs: Motivation

- resource-sensitive, lexicalized framework (cf. CG)
- weakly equivalent to MCFGs
  $\Rightarrow$ appropriate generative capacity for natural language
- inspired by Minimalist syntax (Chomsky 1995)
  $\Rightarrow$ wide empirical coverage &
  formal perspective on linguistic ideas
- efficiently parsable
- MAT-learnable from strings (Stabler & Yoshinaka, in progress)

The Atoms of an MG

Minimalist Grammars (MGs; Stabler 1997)

An MG is a 5-tuple $G := \langle \Sigma, Op, Feat, F, Lex \rangle$, where

- Σ is an alphabet,
- $Op := \{merge, move\}$ is the set of structure-building operations.
- *Feat* is a non-empty finite set of
  - category features f, } trigger *merge*
  - selector features $=f$, } trigger *merge*
  - movement licensee features $-f$, } trigger *move*
  - movement licensor features $+f$, } trigger *move*
- $F \subseteq Feat$ is a set of final category features,
- the lexicon *Lex* is a finite subset of $\Sigma^* \times Feat^+$,

For every MG it suffices to specify *Lex* and *F*.

An MG Example

MG with $F = \{C\}$

men :: N	like :: $= \mathrm{D}\ = \mathrm{D}\ \mathrm{V}$
the :: $= \mathrm{N}\ \mathrm{D}$	ε :: $= \mathrm{V}\ \mathrm{C}$
what :: $\mathrm{D}\ - \mathrm{wh}$	do :: $= \mathrm{V}\ + \mathrm{wh}\ \mathrm{C}$

An MG Example

MG with $F = \{C\}$

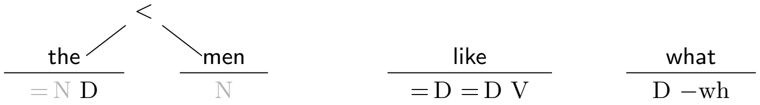
men :: N	like :: $=\mathrm{D}\ =\mathrm{D}\ \mathrm{V}$
the :: $=\mathrm{N}\ \mathrm{D}$	ε :: $=\mathrm{V}\ \mathrm{C}$
what :: $\mathrm{D}\ -\mathrm{wh}$	do :: $=\mathrm{V}\ +\mathrm{wh}\ \mathrm{C}$

$$\frac{\text{the}}{=\mathrm{N}\ \mathrm{D}} \qquad \frac{\text{men}}{\mathrm{N}} \qquad \frac{\text{like}}{=\mathrm{D}\ =\mathrm{D}\ \mathrm{V}} \qquad \frac{\text{what}}{\mathrm{D}\ -\mathrm{wh}}$$

An MG Example

MG with $F = \{C\}$

men :: N	like :: $=\mathrm{D}\ =\mathrm{D}\ \mathrm{V}$
the :: $=\mathrm{N}\ \mathrm{D}$	ε :: $=\mathrm{V}\ \mathrm{C}$
what :: $\mathrm{D}\ -\mathrm{wh}$	do :: $=\mathrm{V}\ +\mathrm{wh}\ \mathrm{C}$



An MG Example

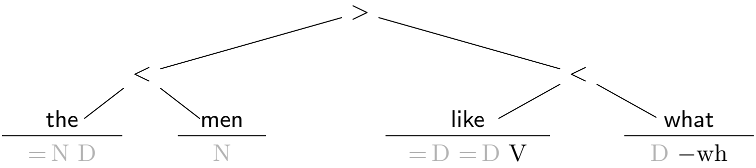
MG with $F = \{C\}$

men :: N	like :: $=\mathrm{D}\ =\mathrm{D}\ \mathrm{V}$
the :: $=\mathrm{N}\ \mathrm{D}$	ε :: $=\mathrm{V}\ \mathrm{C}$
what :: $\mathrm{D}\ -\mathrm{wh}$	do :: $=\mathrm{V}\ +\mathrm{wh}\ \mathrm{C}$

An MG Example

MG with $F = \{C\}$

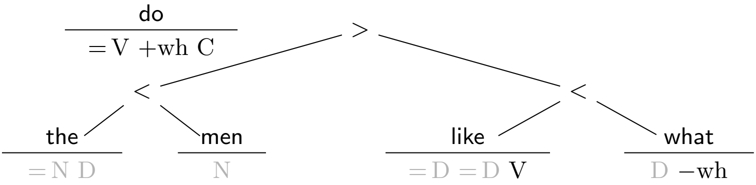
men :: N	like :: $=\mathrm{D}\ =\mathrm{D}\ \mathrm{V}$
the :: $=\mathrm{N}\ \mathrm{D}$	ε :: $=\mathrm{V}\ \mathrm{C}$
what :: $\mathrm{D}\ -\mathrm{wh}$	do :: $=\mathrm{V}\ +\mathrm{wh}\ \mathrm{C}$



An MG Example

MG with $F = \{C\}$

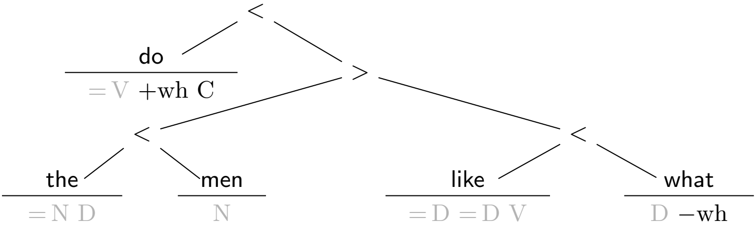
men :: N	like :: $= \mathrm{D} \; = \mathrm{D} \; \mathrm{V}$
the :: $= \mathrm{N} \; \mathrm{D}$	ε :: $= \mathrm{V} \; \mathrm{C}$
what :: $\mathrm{D} \; - \mathrm{wh}$	do :: $= \mathrm{V} \; + \mathrm{wh} \; \mathrm{C}$



An MG Example

MG with $F = \{C\}$

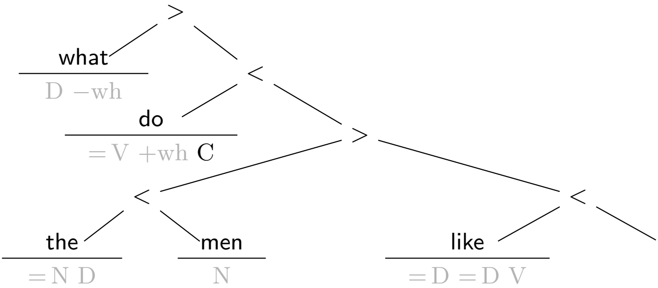
men :: N	like :: $=\mathrm{D}\ =\mathrm{D}\ \mathrm{V}$
the :: $=\mathrm{N}\ \mathrm{D}$	ε :: $=\mathrm{V}\ \mathrm{C}$
what :: $\mathrm{D}\ -\mathrm{wh}$	do :: $=\mathrm{V}\ +\mathrm{wh}\ \mathrm{C}$



An MG Example

MG with $F = \{C\}$

men :: N	like :: $=\mathrm{D}\ =\mathrm{D}\ \mathrm{V}$
the :: $=\mathrm{N}\ \mathrm{D}$	ε :: $=\mathrm{V}\ \mathrm{C}$
what :: $\mathrm{D}\ -\mathrm{wh}$	do :: $=\mathrm{V}\ +\mathrm{wh}\ \mathrm{C}$



The Shortest Move Constraint (SMC)

Shortest Move Constraint (SMC; informal definition)

At no point in a derivation may two lexical items exhibit the same licensee feature as their first unchecked feature.

Example of an Illicit Configuration

what	like	what
$D\ -wh$	$=D\ =D\ V$	$D\ -wh$

The Shortest Move Constraint (SMC)

Shortest Move Constraint (SMC; informal definition)

At no point in a derivation may two lexical items exhibit the same licensee feature as their first unchecked feature.

Example of an Illicit Configuration

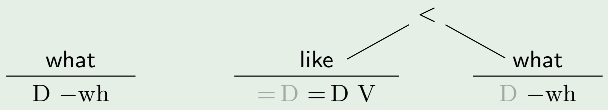
$$\frac{\text{what}}{\mathrm{D}\ -\mathrm{wh}} \qquad \frac{\text{like}}{=\mathrm{D}\ =\mathrm{D}\ \mathrm{V}} \qquad \frac{\text{what}}{\mathrm{D}\ -\mathrm{wh}}$$

The Shortest Move Constraint (SMC)

Shortest Move Constraint (SMC; informal definition)

At no point in a derivation may two lexical items exhibit the same licensee feature as their first unchecked feature.

Example of an Illicit Configuration

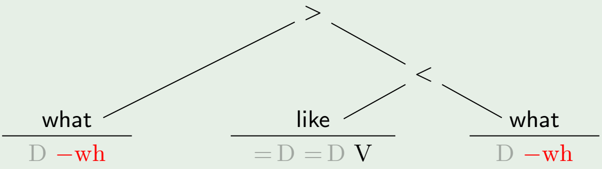


The Shortest Move Constraint (SMC)

Shortest Move Constraint (SMC; informal definition)

At no point in a derivation may two lexical items exhibit the same licensee feature as their first unchecked feature.

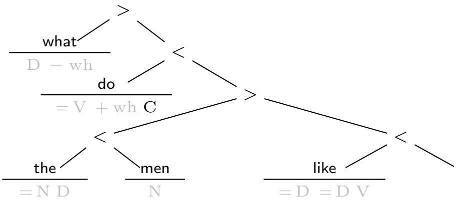
Example of an Illicit Configuration

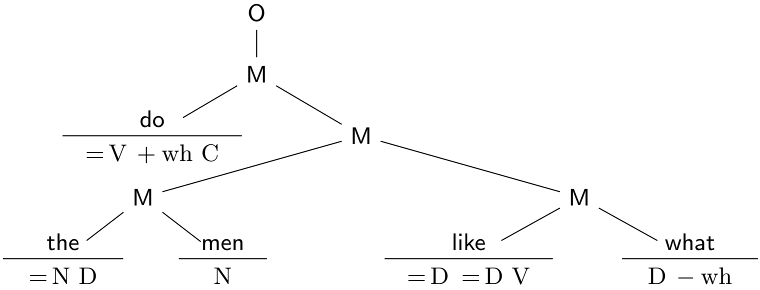


Derivation Trees

Useful Fact

Every MG is fully specified by its set of derivation trees, which is regular (Michaelis 1998).





Defining Derivation Trees: Slices and Occurrences

Goal A tree geometric definition of well-formed derivations

Idea Lexical items are "tree atoms" that can be combined to form derivation trees, but certain constraints hold.

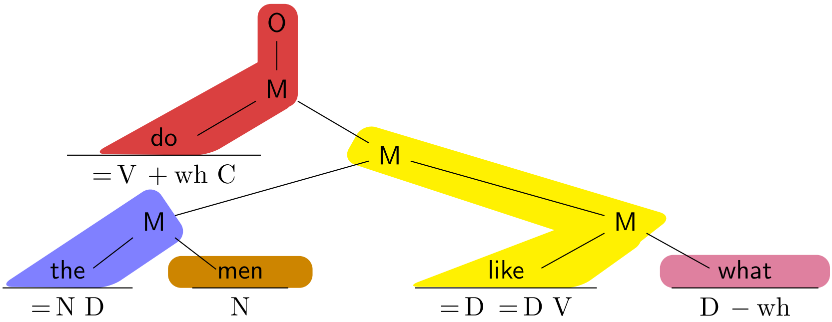
Slices (Intuitive Definition)

Slices are the **derivation tree equivalent of phrasal projection**: A slice marks the subpart of the derivation that a lexical item has control over by virtue of its selector and licensor features.

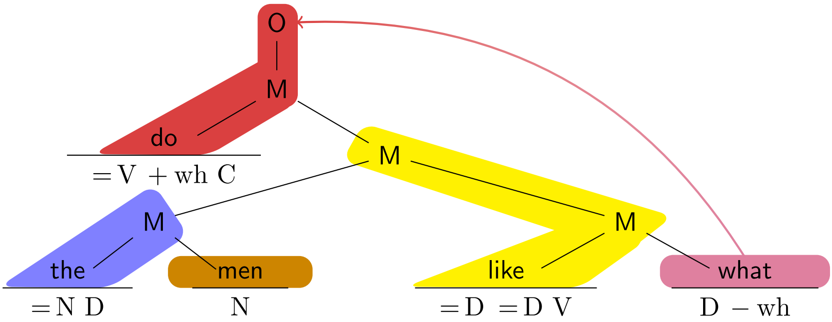
Occurrences

The **occurrences** of a lexical item are those movement nodes that erased one of its licensee features.

Example of Slices & Occurrences



Example of Slices & Occurrences

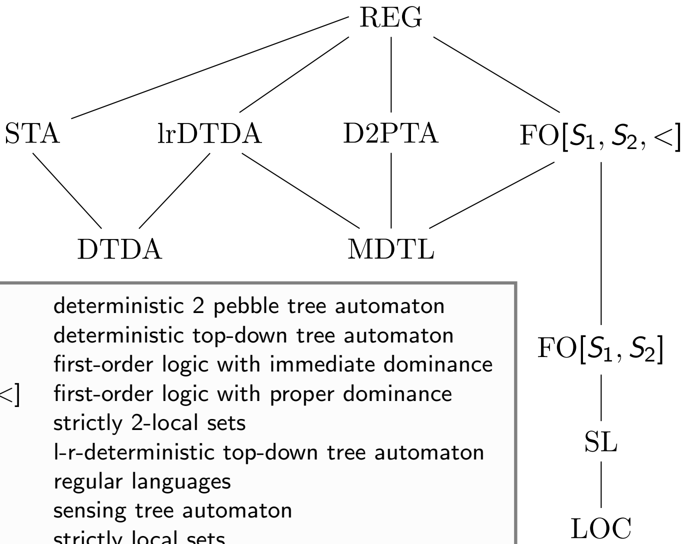


MDTLs as Combinations of Slices

The **Minimalist derivation tree language** (MDTL) of MG G is the largest set of combinations of slices such that (all $l \in Lex_G$)

- *Final*: The root belongs to the slice of some l such that l has a final category feature.
- *Merge*: If the n-th feature of l is $=f$, then the n-th mother of l immediately dominates the slice of some l' such that l' has category feature f.
- *Move*:
    - For every movement node n there is exactly one lexical item l such that n is an occurrence of l.
    - For every licensee feature of some lexical item l, there is exactly one movement node that is an occurrence of l.

Unrestricted MDTLs in the Subregular Hierarchy



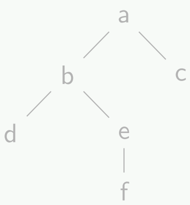
D2PTA	deterministic 2 pebble tree automaton
DTDA	deterministic top-down tree automaton
$FO[S_1, S_2]$	first-order logic with immediate dominance
$FO[S_1, S_2, <]$	first-order logic with proper dominance
LOC	strictly 2-local sets
lrDTDA	l-r-deterministic top-down tree automaton
REG	regular languages
STA	sensing tree automaton
SL	strictly local sets

Example: Undefinability in SL

Strictly Local Languages

A tree language L is strictly k-local if there is some finite set S of subtrees of depth $d \leq k$ such that L is the smallest set containing all trees whose k-factors belong to S.

Tree and its 3-Factors

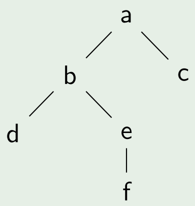


Example: Undefinability in SL

Strictly Local Languages

A tree language L is strictly k-local if there is some finite set S of subtrees of depth $d \leq k$ such that L is the smallest set containing all trees whose k-factors belong to S.

Tree and its 3-Factors

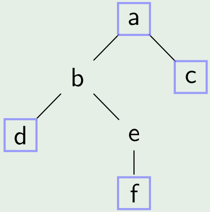


Example: Undefinability in SL

Strictly Local Languages

A tree language L is strictly k-local if there is some finite set S of subtrees of depth $d \leq k$ such that L is the smallest set containing all trees whose k-factors belong to S.

Tree and its 3-Factors

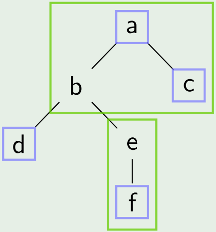


Example: Undefinability in SL

Strictly Local Languages

A tree language L is strictly k-local if there is some finite set S of subtrees of depth $d \leq k$ such that L is the smallest set containing all trees whose k-factors belong to S.

Tree and its 3-Factors

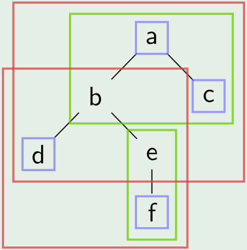


Example: Undefinability in SL

Strictly Local Languages

A tree language L is strictly k-local if there is some finite set S of subtrees of depth $d \leq k$ such that L is the smallest set containing all trees whose k-factors belong to S.

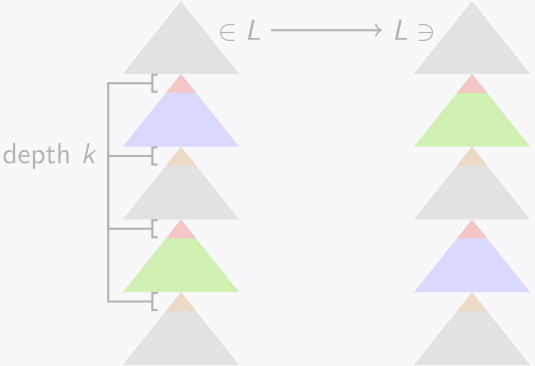
Tree and its 3-Factors



Example: Undefinability in SL (Cont.)

- If a tree language is strictly local, then it must be closed under k-guarded vertical swaps for some $k \in \mathbb{N}$. MDTLs aren't.

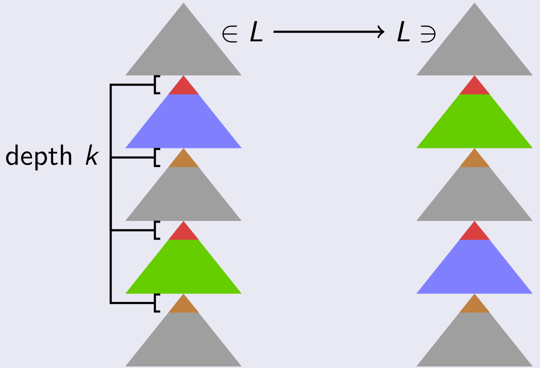
Closure Under k-Guarded Vertical Swaps



Example: Undefinability in SL (Cont.)

- If a tree language is strictly local, then it must be closed under k-guarded vertical swaps for some $k \in \mathbb{N}$. MDTLs aren't.

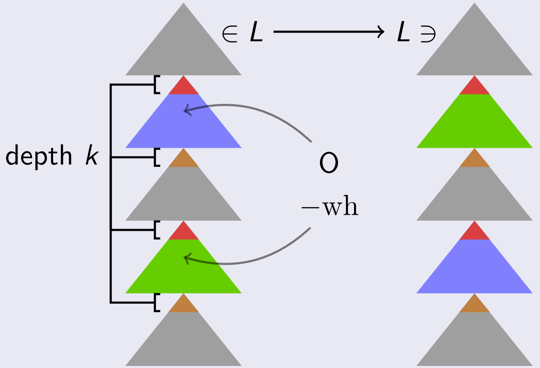
Closure Under k-Guarded Vertical Swaps



Example: Undefinability in SL (Cont.)

- If a tree language is strictly local, then it must be closed under k-guarded vertical swaps for some $k \in \mathbb{N}$. MDTLs aren't.

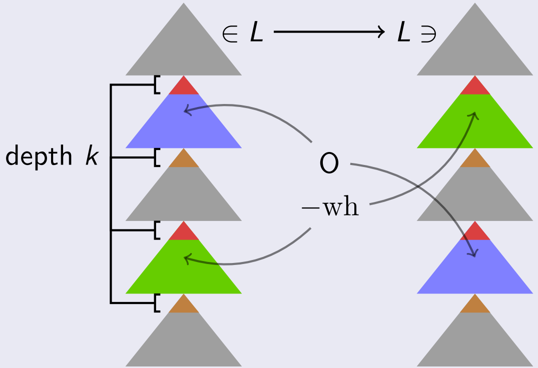
Closure Under k-Guarded Vertical Swaps



Example: Undefinability in SL (Cont.)

- If a tree language is strictly local, then it must be closed under k-guarded vertical swaps for some $k \in \mathbb{N}$. MDTLs aren't.

Closure Under k-Guarded Vertical Swaps



What to Learn from the Results

- **Incomparable with local sets**
  Merge by itself already establishes dependencies that extend beyond trees of depth 1.
- **(Not) recognizable by top-down tree automata**
  The lack of meaningful non-terminal symbols makes MDTLs highly non-deterministic from a top-down perspective.
  The automaton must be capable of unbounded look-ahead into one subtree of the root.
- **Undefinability in $\mathrm{FO}[S_1, S_2]$/Definability in $\mathrm{FO}[S_1, S_2, <]$**
  While the conditions *Final* and *Merge* are local, movement dependencies are unbounded.
- **Recognizability by deterministic 2 pebble tree automata**
  Well-formedness conditions are not co-dependent.
  Movement nodes can be checked independently despite them being closely related via the notion of occurrence.

Importance of the SMC

- Well-known: SMC crucial in rendering MDTLs regular.
- But all the previous definability results also hinge on the SMC.

The Effect of the SMC

Occurrences can be computed from path conditions:

- Given: lexical item l with licensee features $-f_1, \ldots, -f_n$.
- $occ_1(l) :=$ the first O-node properly dominating the slice of l with feature $+f_1$.
- $occ_n(l) :=$ the first O-node properly dominating $occ_{n-1}(l)$ with feature $+f_n$.

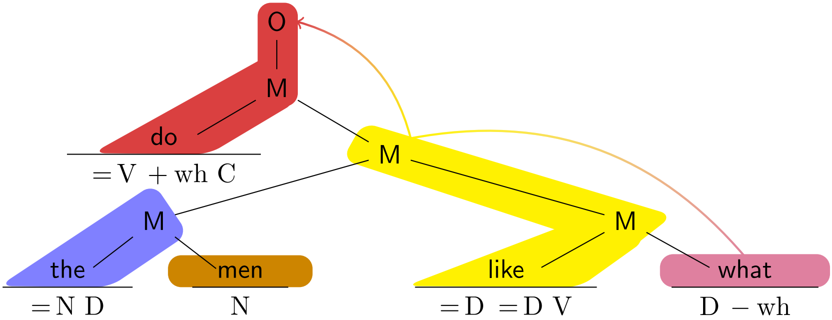
Insight Even though Move is unbounded, it is still structurally simple, thanks to the SMC

Introducing Locality

Most undefinability results are due to the unboundedness of Move.
So what if we limit how far an element may be moved in one step?

k-Locality

An MDTL L is k-local if it holds for all derivations $d \in L$ and lexical items $l \in d$ that at most $k-1$ slices intervene between the occurrences of l (counting l itself as an occurrence).



MDTLs of *k*-Local MGs are Strictly Local

Theorem (*k*-Local → Strictly κ-Local)

For every k-local MG, its MDTL is strictly κ-local, where

- $\kappa = (|\gamma| + 1) * (|\delta| * k + 1) + 1$,
- $|\gamma|$ *is the maximum of licensor and selector features on a lexical item,*
- $|\delta|$ *is the maximum of licensee features on a lexical item.*

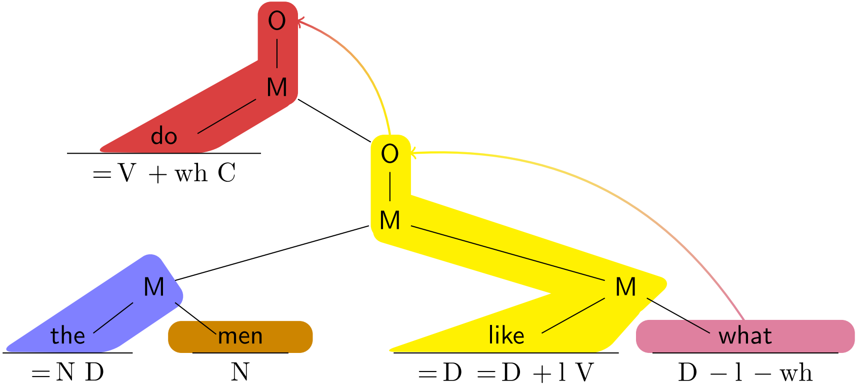
Proof.

- Both Merge and Move are local operations in a *k*-local MG.
- All their requirements must be satisfied in some local domain of bounded size $s \leq \kappa$.
- So if every subtree of depth κ is well-formed, the entire tree is, too.
- This implies being strictly κ-local. □

k-locality = 1-locality

Lemma (Locality Reduction)

For every k-local MG there is a weakly equivalent 1-local MG.



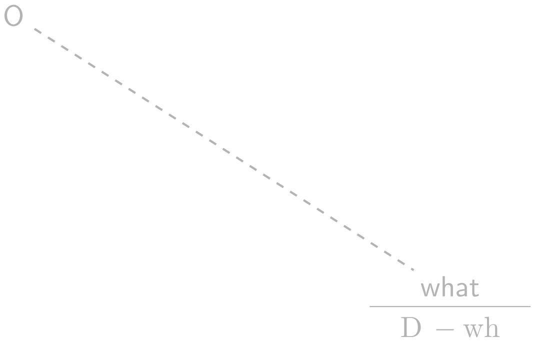
An Unexpected Problem?

- **Potential problem**
  MGs have lexicons of finite size, and each lexical item carries only a finite number of features
- **Why the procedure works**
  - SMC ⇒ upper bound on number of moving elements
  - *k*-locality ⇒ length of movement steps bounded
  - Hence only a finite number of new features is needed.
- **Technical remark**
  Procedure implemented by a linear tree transducer
  ⇒ output is indeed an MDTL (Graf 2011; Kobele 2011)

Localizing Unrestricted MDTLs

Problem *k*-locality does not hold for unrestricted MDTLs.

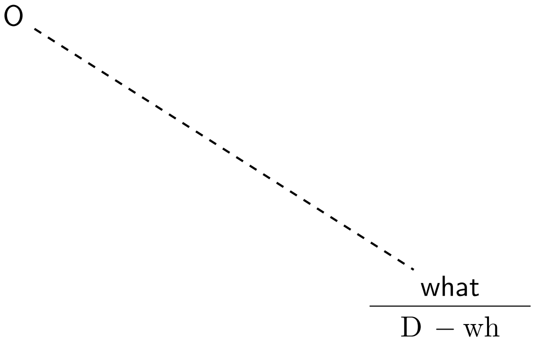
New idea Rather than adding new features, introduce new moving elements on which the original mover can piggy-back.



Localizing Unrestricted MDTLs

Problem *k*-locality does not hold for unrestricted MDTLs.

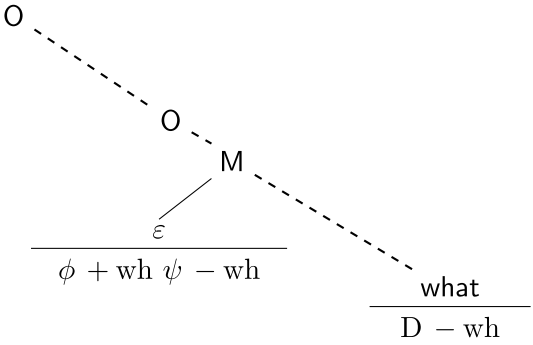
New idea Rather than adding new features, introduce new moving elements on which the original mover can piggy-back.



Localizing Unrestricted MDTLs

Problem k-locality does not hold for unrestricted MDTLs.

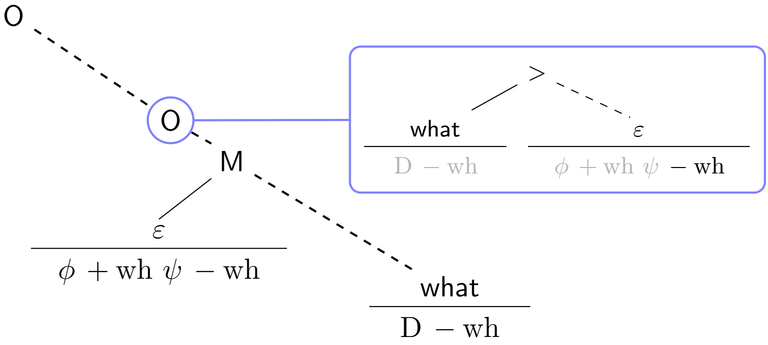
New idea Rather than adding new features, introduce new moving elements on which the original mover can piggy-back.



Localizing Unrestricted MDTLs

Problem *k*-locality does not hold for unrestricted MDTLs.

New idea Rather than adding new features, introduce new moving elements on which the original mover can piggy-back.



Unrestricted MG ≡ 1-local MG

Theorem (Unrestricted/Local Equivalence)

For every unrestricted MG there exists a 1-local MG yielding the same string language (and almost the same tree language).

Technical remarks

- New features needed for remnant movement
- Careful bookkeeping required (which features to add/remove, where to insert new lexical items)
- Works only because of SMC
- Nonetheless computable by linear tree transducer
  $\Rightarrow$ output is an MDTL

Unrestricted MG $\equiv$ 1-local MG

Theorem (Unrestricted/Local Equivalence)

For every unrestricted MG there exists a 1-local MG yielding the same string language (and almost the same tree language).

Technical remarks

- New features needed for remnant movement
- Careful bookkeeping required (which features to add/remove, where to insert new lexical items)
- Works only because of SMC
- Nonetheless computable by linear tree transducer
  $\Rightarrow$ output is an MDTL

Conclusion & Discussion

- Local MGs are strictly local, unrestricted MGs are not.
- The non-locality of the latter follows from the unboundedness of Move.
- Still, both Merge and Move are structurally simple operations, thanks to the SMC.
- Moreover, every unrestricted MG can be localized.
- Status of locality conditions in linguistic theories?

References

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

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.

Kobele, Gregory M. 2011. Minimalist tree languages are closed under intersection with recognizable tree languages. To appear in Proceedings of LACL2011.

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.

Kolb, Hans-Peter, Jens Michaelis, Uwe Mönnich, and Frank Morawietz. 2003. An operational and denotational approach to non-context-freeness. *Theoretical Computer Science* 293:261–289.

Michaelis, Jens. 1998. Derivational minimalism is mildly context-sensitive. *Lecture Notes in Artificial Intelligence* 2014:179–198.

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.