Closure Properties of Minimalist Derivation Tree Languages

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

University of California, Los Angeles

LACL 2011
June 30, 2011

Outline

Regular = Recognized by Bottom-Up Tree Automaton

- **Bottom-up tree automata** generalize finite-state automata from strings to trees.
- Only significant **change in the transition function**: domain extended from pairs of symbols and states to $n+1$ tuples $\langle q_1, \ldots, q_n, \sigma^{(n)} \rangle$, where $\sigma^{(n)}$ is a symbol of arity $n \geq 0$.

Deterministic Bottom-Up Tree Automata

A *deterministic bottom-up tree automaton* is a 4-tuple $A := \langle \Sigma, Q, F, \delta \rangle$, where

- Σ is a ranked alphabet,
- Q is a finite set of states (i.e. of unary symbols $q \notin \Sigma$),
- $F \subseteq Q$ is the set of final states,
- $\delta \colon \left(\bigcup_{n \geq 0} Q^n \times \Sigma^{(n)} \right) \to Q$ is the transition function.

ODD: A Regular Tree Language

Let *ODD* be the language of all (at most) binary branching trees over alphabet $\Sigma := \{a^{(0)}, a^{(1)}, a^{(2)}\}$ such that **every tree has an odd number of nodes**.

Automaton for *ODD*

$A_{ODD} := \langle \{a^{(0)}, a^{(1)}, a^{(2)}\}, \{O, E\}, \{O\}, \delta \rangle$,
where δ is given by the following rules:

$$\begin{array}{rr} a \to O & (O, O, a) \to O \\ (O, a) \to E & (O, E, a) \to E \\ (E, a) \to O & (E, O, a) \to E \\ & (E, E, a) \to O \end{array}$$

ODD: A Regular Tree Language

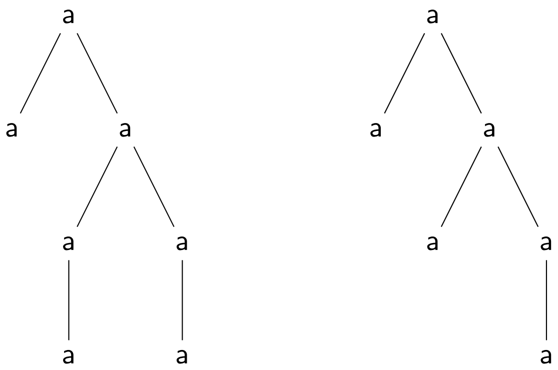
Let *ODD* be the language of all (at most) binary branching trees over alphabet $\Sigma := \{a^{(0)}, a^{(1)}, a^{(2)}\}$ such that **every tree has an odd number of nodes**.

Automaton for *ODD*

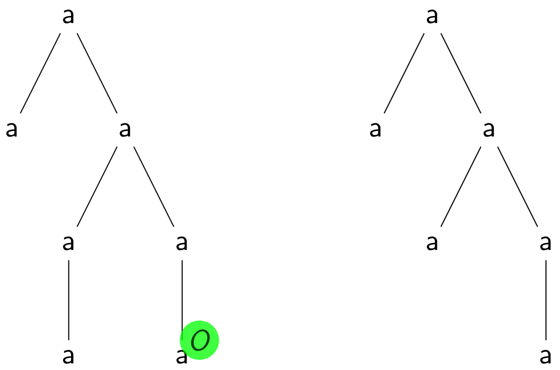
$A_{ODD} := \langle \{a^{(0)}, a^{(1)}, a^{(2)}\}, \{O, E\}, \{O\}, \delta \rangle$,
where δ is given by the following rules:

$$\begin{array}{rr} a \rightarrow O & (O, O, a) \rightarrow O \\ (O, a) \rightarrow E & (O, E, a) \rightarrow E \\ (E, a) \rightarrow O & (E, O, a) \rightarrow E \\ & (E, E, a) \rightarrow O \end{array}$$

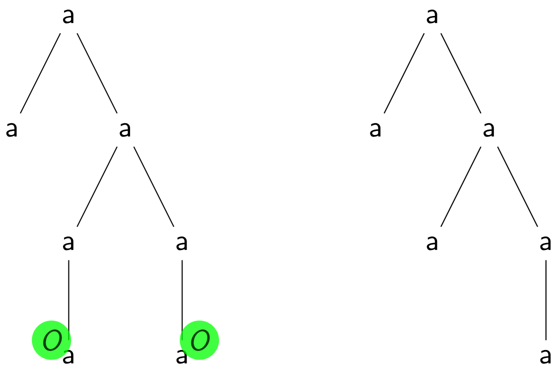
Two Runs of A_{ODD}



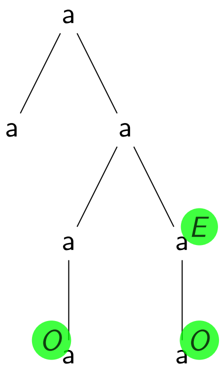
Two Runs of A_{ODD}

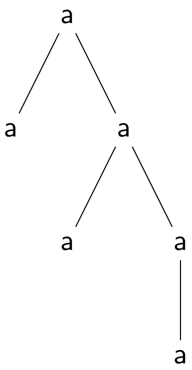


Two Runs of A_{ODD}

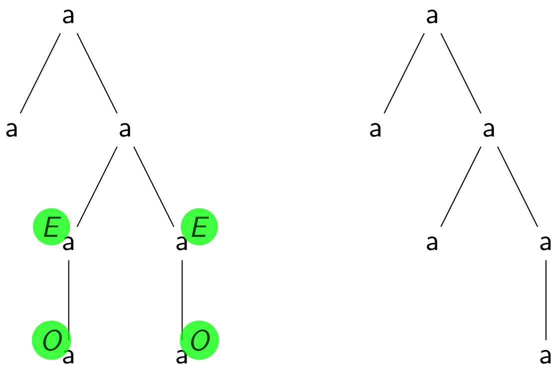


Two Runs of A_{ODD}

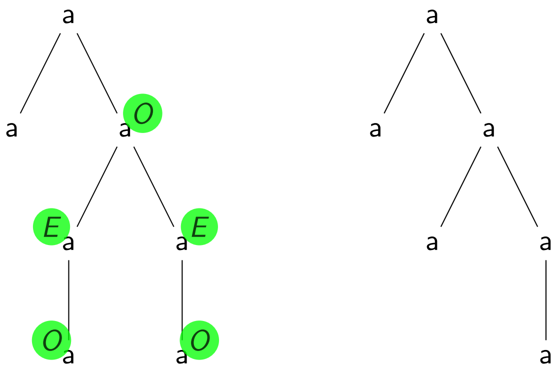




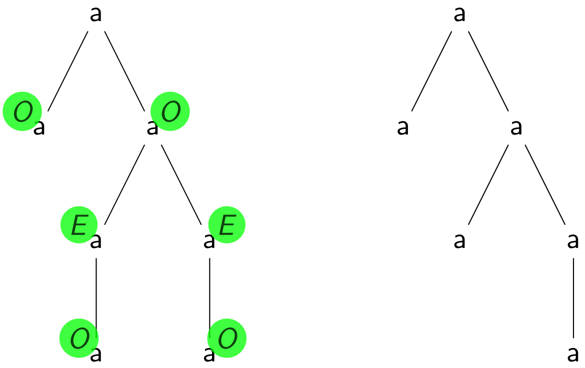
Two Runs of A_{ODD}



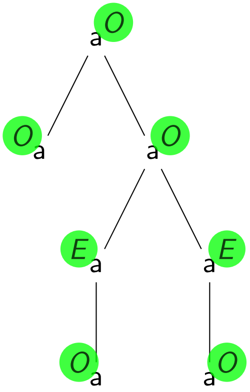
Two Runs of A_{ODD}

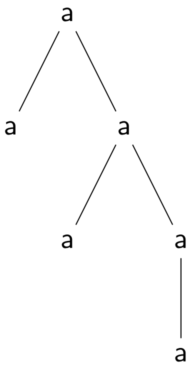


Two Runs of A_{ODD}

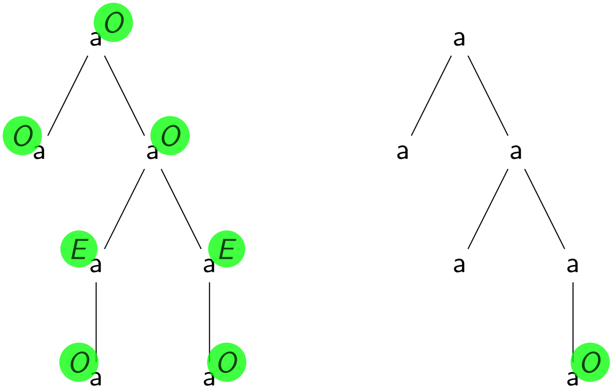


Two Runs of A_{ODD}

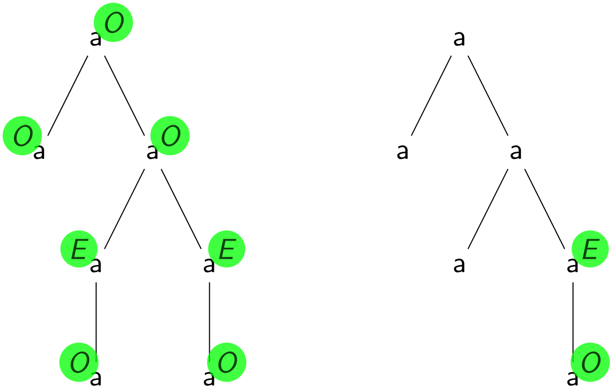




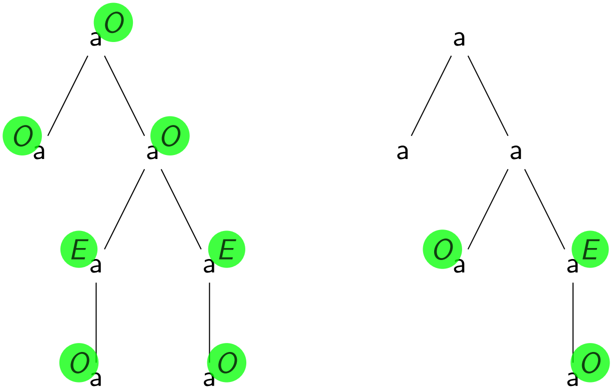
Two Runs of A_{ODD}



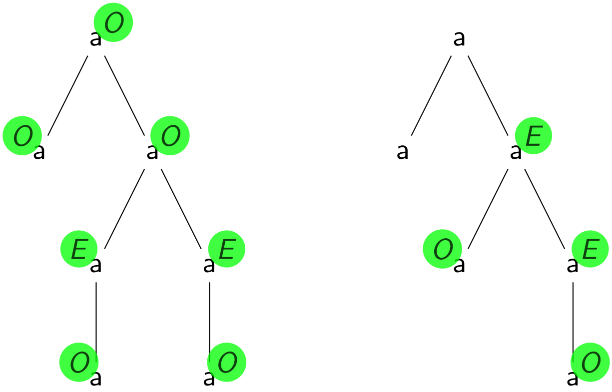
Two Runs of A_{ODD}



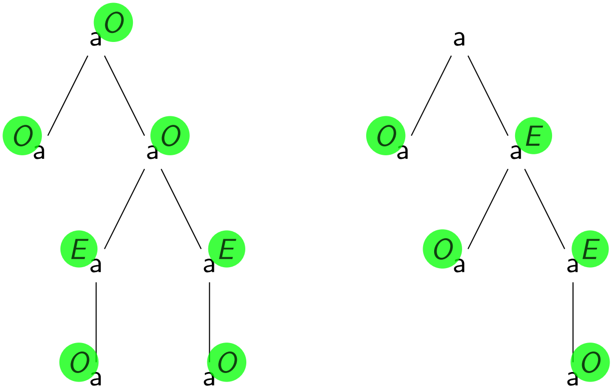
Two Runs of A_{ODD}



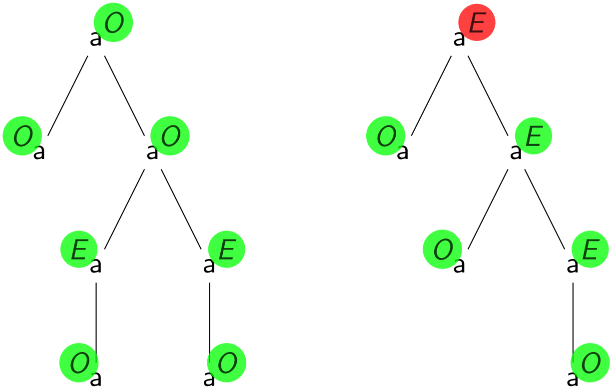
Two Runs of A_{ODD}



Two Runs of A_{ODD}



Two Runs of A_{ODD}



Regular Tree Languages/Automata for Linguistics

- Just like regular string languages, regular tree languages are very well-behaved mathematically
  $\Rightarrow$ attractive from a computational perspective
- Almost all parts of Government-and-Binding theory can be expressed by bottom-up automata (Rogers 1998)
  $\Rightarrow$ regular tree languages sufficiently powerful for most syntactic generalizations/constraints
- But the string yield of a regular tree language is context-free
  $\Rightarrow$ too weak for natural language
- Minimalist grammars (MGs) generate MCFLs, yet can be fully specified by regular tree languages. But is it possible to **add regular constraints to MGs without increasing their weak generative capacity?**

The Atoms of a Minimalist Grammar

Minimalist Grammars (MGs; Stabler 1997)

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

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

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

Bare Phrase Structure Trees

- My definition of *merge* and *move* is tree-based.
- It builds on the notion of Bare Phrase Structure trees and Headedness.

Extended Lexicon

Given a lexicon *Lex*, its *extended lexicon Elex* is the smallest set such that, for $\sigma \in \Sigma^*$, $f \in Feat$, and $\delta \in Feat^*$

- $l \in Lex \rightarrow l \in Elex$
- $l := \langle \sigma, f\delta \rangle \in Elex \rightarrow l' := \langle \sigma, \delta \rangle \in Elex$

Bare Phrase Structure Trees (BPS Trees)

The set of *BPS trees over Elex* consists of all strictly binary branching trees over the ranked alphabet $\{<^{(2)}, >^{(2)}\} \cup \{l^{(0)} \mid l \in Elex\}$.

Bare Phrase Structure Trees

- My definition of *merge* and *move* is tree-based.
- It builds on the notion of Bare Phrase Structure trees and Headedness.

Extended Lexicon

Given a lexicon *Lex*, its *extended lexicon Elex* is the smallest set such that, for $\sigma \in \Sigma^*$, $f \in Feat$, and $\delta \in Feat^*$

- $l \in Lex \rightarrow l \in Elex$
- $l := \langle \sigma, f\delta \rangle \in Elex \rightarrow l' := \langle \sigma, \delta \rangle \in Elex$

Bare Phrase Structure Trees (BPS Trees)

The set of *BPS trees over Elex* consists of all strictly binary branching trees over the ranked alphabet $\{<^{(2)}, >^{(2)}\} \cup \{l^{(0)} \mid l \in Elex\}$.

Bare Phrase Structure Trees

- My definition of *merge* and *move* is tree-based.
- It builds on the notion of Bare Phrase Structure trees and Headedness.

Extended Lexicon

Given a lexicon *Lex*, its *extended lexicon Elex* is the smallest set such that, for $\sigma \in \Sigma^*$, $f \in Feat$, and $\delta \in Feat^*$

- $l \in Lex \rightarrow l \in Elex$
- $l := \langle \sigma, f\delta \rangle \in Elex \rightarrow l' := \langle \sigma, \delta \rangle \in Elex$

Bare Phrase Structure Trees (BPS Trees)

The set of *BPS trees over Elex* consists of all strictly binary branching trees over the ranked alphabet $\left\{ <^{(2)}, >^{(2)} \right\} \cup \left\{ l^{(0)} \mid l \in Elex \right\}$.

Headedness

Headedness

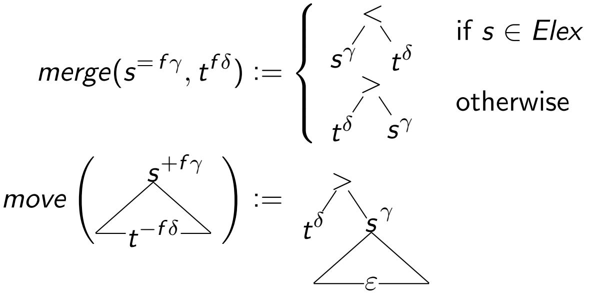
Given a BPS tree t, the *head* of t is given by

$$head(t) := \begin{cases} t & \text{if } t \in Elex \\ head(t_1) & \text{if } t := \underset{t_1 \quad t_2}{<} \\ head(t_2) & \text{if } t := \underset{t_1 \quad t_2}{>} \end{cases}$$

Notation t^δ denotes that $head(t)$ carries feature string δ

Defining Merge & Move

Let $\gamma, \delta \in \mathit{Feat}^*$.



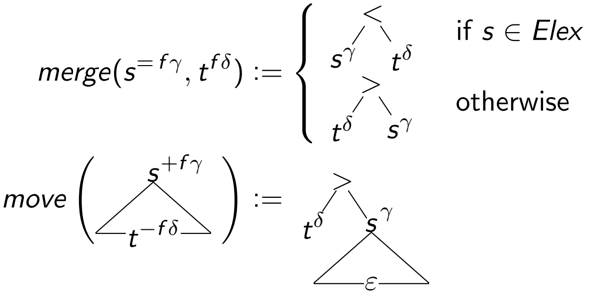
Shortest Move Constraint (SMC)

Every tree $s^{+f\gamma}$ in the domain of *move* has exactly one subtree t such that the first feature of $\mathit{head}(t)$ is $-f$.

Thanks to the SMC, **both Merge and Move are deterministic**.

Defining Merge & Move

Let $\gamma, \delta \in \mathit{Feat}^*$.



Shortest Move Constraint (SMC)

Every tree $s^{+f\gamma}$ in the domain of *move* has exactly one subtree t such that the first feature of $head(t)$ is $-f$.

Thanks to the SMC, **both Merge and Move are deterministic**.

Derived Tree Language & Expressivity

Derived Tree Language

The tree language $L(G)$ derived by MG G with lexicon Lex_G is the largest set of BPS trees such that

- $L(G) \subseteq closure(Lex_G, \{merge, move\})$,
- for every $t \in L(G)$, there is some $f \in F_G$ such that the feature component of $head(t)$ consists only of f,
- all other leaves have an empty feature component.

Generated String Language

The string language generated by MG G is the string yield of $L(G)$.

Theorem (Harkema 2001; Michaelis 1998, 2001)

MCFGs and MGs are weakly equivalent.

Derived Tree Language & Expressivity

Derived Tree Language

The tree language $L(G)$ derived by MG G with lexicon Lex_G is the largest set of BPS trees such that

- $L(G) \subseteq closure(Lex_G, \{merge, move\})$,
- for every $t \in L(G)$, there is some $f \in F_G$ such that the feature component of $head(t)$ consists only of f,
- all other leaves have an empty feature component.

Generated String Language

The string language generated by MG G is the string yield of $L(G)$.

Theorem (Harkema 2001; Michaelis 1998, 2001)

MCFGs and MGs are weakly equivalent.

Derived Tree Language & Expressivity

Derived Tree Language

The tree language $L(G)$ derived by MG G with lexicon Lex_G is the largest set of BPS trees such that

- $L(G) \subseteq closure(Lex_G, \{merge, move\})$,
- for every $t \in L(G)$, there is some $f \in F_G$ such that the feature component of $head(t)$ consists only of f,
- all other leaves have an empty feature component.

Generated String Language

The string language generated by MG G is the string yield of $L(G)$.

Theorem (Harkema 2001; Michaelis 1998, 2001)

MCFGs and MGs are weakly equivalent.

A Toy Example (Without Recursion)

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 :: $=\mathrm{V}\ +\mathrm{wh}\ \mathrm{C}$

A Toy Example (Without Recursion)

MG with $F = \{C\}$

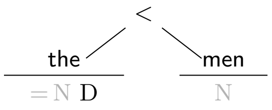
men :: N	like :: $=\mathrm{D}\ =\mathrm{D}\ \mathrm{V}$
the :: $=\mathrm{N}\ \mathrm{D}$	ε :: $=\mathrm{V}\ \mathrm{C}$
what :: $\mathrm{D}\ -\mathrm{wh}$	do :: $=\mathrm{V}\ +\mathrm{wh}\ \mathrm{C}$

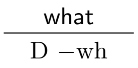
$$\frac{\text{the}}{=\mathrm{N}\ \mathrm{D}} \qquad \frac{\text{men}}{\mathrm{N}} \qquad \frac{\text{like}}{=\mathrm{D}\ =\mathrm{D}\ \mathrm{V}} \qquad \frac{\text{what}}{\mathrm{D}\ -\mathrm{wh}}$$

A Toy Example (Without Recursion)

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 :: $=\mathrm{V}\ +\mathrm{wh}\ \mathrm{C}$

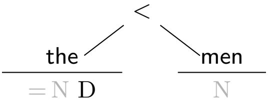


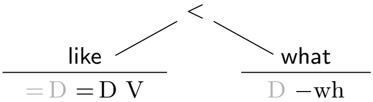


A Toy Example (Without Recursion)

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 :: $=\mathrm{V}\ +\mathrm{wh}\ \mathrm{C}$

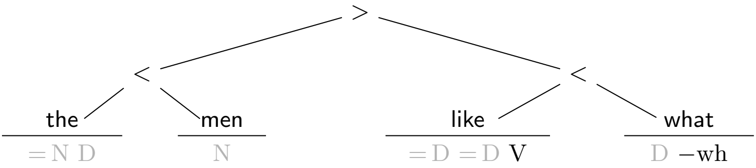




A Toy Example (Without Recursion)

MG with $F = \{C\}$

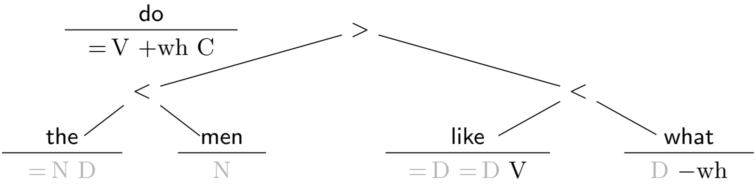
men :: N	like :: $=\mathrm{D}\ =\mathrm{D}\ \mathrm{V}$
the :: $=\mathrm{N}\ \mathrm{D}$	ε :: $=\mathrm{V}\ \mathrm{C}$
what :: $\mathrm{D}\ -\mathrm{wh}$	do :: $=\mathrm{V}\ +\mathrm{wh}\ \mathrm{C}$



A Toy Example (Without Recursion)

MG with $F = \{C\}$

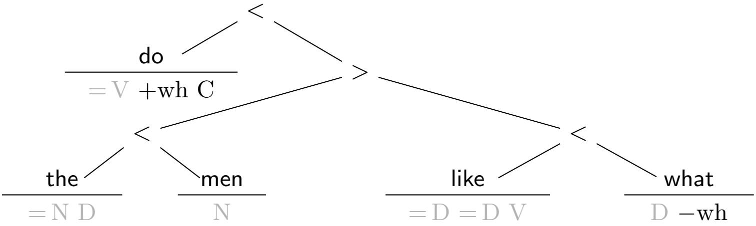
men :: N	like :: $= \mathrm{D} \ = \mathrm{D} \ \mathrm{V}$
the :: $= \mathrm{N} \ \mathrm{D}$	ε :: $= \mathrm{V} \ \mathrm{C}$
what :: $\mathrm{D} \ - \mathrm{wh}$	do :: $= \mathrm{V} \ + \mathrm{wh} \ \mathrm{C}$



A Toy Example (Without Recursion)

MG with $F = \{C\}$

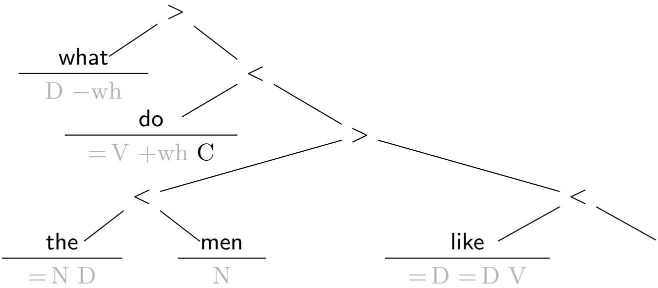
men :: N	like :: $=\mathrm{D}\ =\mathrm{D}\ \mathrm{V}$
the :: $=\mathrm{N}\ \mathrm{D}$	ε :: $=\mathrm{V}\ \mathrm{C}$
what :: $\mathrm{D}\ -\mathrm{wh}$	do :: $=\mathrm{V}\ +\mathrm{wh}\ \mathrm{C}$



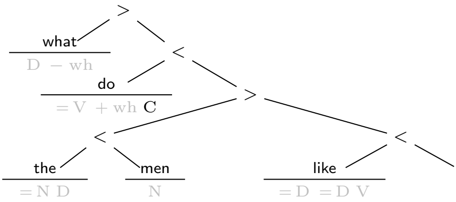
A Toy Example (Without Recursion)

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 :: $=\mathrm{V}\ +\mathrm{wh}\ \mathrm{C}$

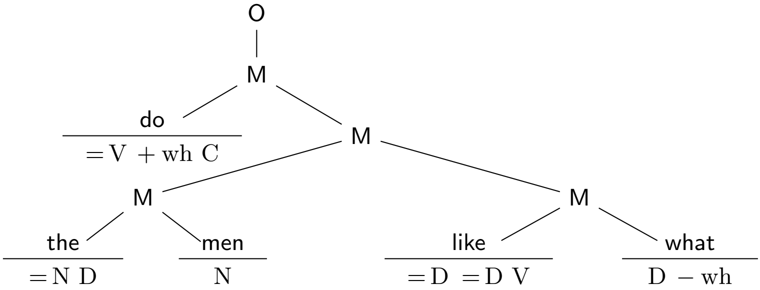


Derivation Trees



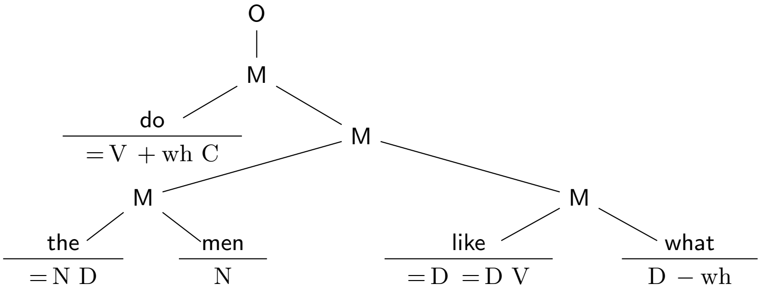
Useful Fact

Every MG is fully specified by its set of derivation trees, which is regular (Kobele et al. 2007).



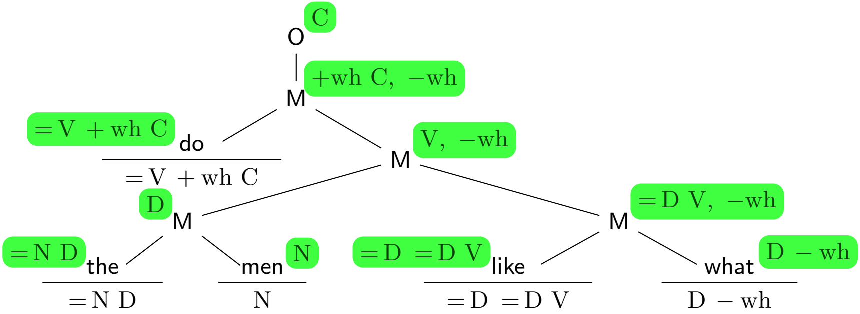
Defining Derivation Trees: The Intuition

- Defining well-formed derivation trees of MG G only requires keeping track of the feature calculus $\Rightarrow$ deterministic bottom-up automaton with **sequences of feature strings as states** (and $F_A := \{\langle f \rangle \mid f \in F_G\}$)
- Due to the SMC, the number of feature strings per state is bounded $\Rightarrow$ finite number of states



Defining Derivation Trees: The Intuition

- Defining well-formed derivation trees of MG G only requires keeping track of the feature calculus
  $\Rightarrow$ deterministic bottom-up automaton with **sequences of feature strings as states** (and $F_A := \{\langle f \rangle \mid f \in F_G\}$)
- Due to the SMC, the number of feature strings per state is bounded $\Rightarrow$ finite number of states



Non-Closure Under Intersection with REG

Theorem

The class of MDTLs is not closed under intersection with regular tree languages.

Proof.

- Let *ODD* contain all trees with an odd number of nodes.
- Let G be the MG given by $F_G = \{c\}$ and Lex_G:
  a :: a b :: $= \mathrm{a} = \mathrm{a} + \mathrm{k}\ \mathrm{a}$ c :: $= \mathrm{a}\ \mathrm{c}$
  a :: $\mathrm{a} \ - \mathrm{k}$
- Then there are derivation trees s and t in the closure of Lex_G under $\{merge, move\}$ that both end in a final category and contain the same lexical items. It is easy to see that $s \in mder(G')$ iff $t \in mder(G')$ for any MG G', yet $s \notin mder(G) \cap ODD \ni t$. □

Non-Closure Under Intersection with REG

Theorem

The class of MDTLs is not closed under intersection with regular tree languages.

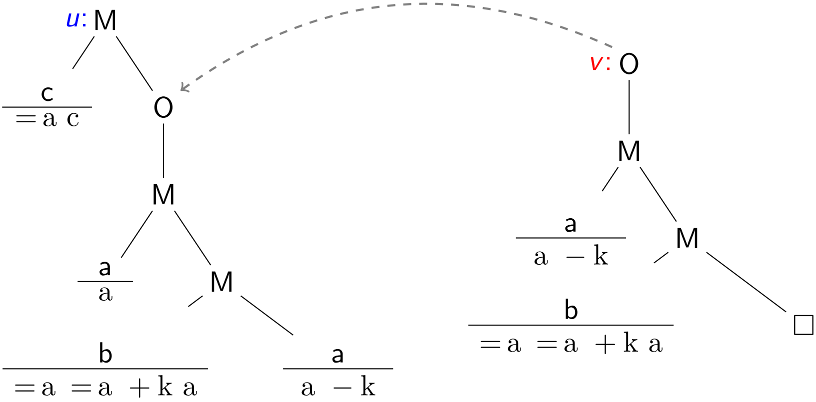
Proof.

- Let ODD contain all trees with an odd number of nodes.
- Let G be the MG given by $F_G = \{c\}$ and Lex_G:

  a :: a b :: $=\mathrm{a}\ =\mathrm{a}\ +\mathrm{k}\ \mathrm{a}$ c :: $=\mathrm{a}\ \mathrm{c}$

  a :: $\mathrm{a}\ -\mathrm{k}$
- Then there are derivation trees s and t in the closure of Lex_G under $\{merge, move\}$ that both end in a final category and contain the same lexical items. It is easy to see that $s \in mder(G')$ iff $t \in mder(G')$ for any MG G', yet $s \notin mder(G) \cap ODD \ni t$. □

Choice of s and t



$$s = u \notin ODD$$
$$t = u + v \in ODD$$

Defining P-Closure

Projection

Let $\lambda : \Sigma \to \Omega$ be a many-to-one map between alphabets, and π its extension from alphabets to trees.
Tree t is a *projection* of s iff there is a π such that $t = \pi(s)$.
The notion extends to tree languages in the natural way.

P[rojection]-Closure

Given a class of languages $\mathcal{L}$ and an operation O, $\mathcal{L}$ is *p-closed* under O iff the result of applying O to some $L \in \mathcal{L}$ is a projection of some $L' \in \mathcal{L}$.

P-Closure Under Intersection with REG

Theorem (REG Intersection P-Closure)

The class of MDTLs over alphabet Σ and features Feat is p-closed under intersection with regular tree languages.

Outline of Proof

- Inspired by Thatcher's theorem (translate recognizable sets into local ones by incorporating states into alphabet)
- Crux: Internal node labels of a derivation tree cannot be refined $\Rightarrow$ *slices* as a way of relating interior nodes to features on lexical items
- Procedure for refining category and selector features so that they incorporate states of the deterministic bottom-up automaton recognizing regular language

Slices

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

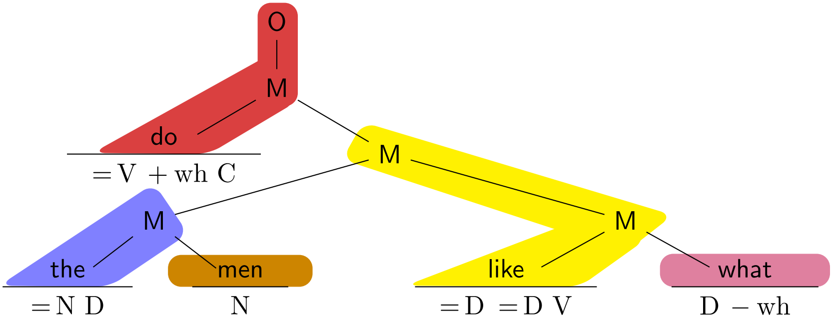
Slices

Given a derivation tree t and lexical item l occurring in t, $\mathrm{slice}(l)$ is defined as follows:

- $l \in \mathrm{slice}(l)$,
- if node n of t immediately dominates a node $s \in \mathrm{slice}(l)$, then $n \in \mathrm{slice}(l)$ iff the operation denoted by the label of n erased a selector or licensor feature of l.

The unique $n \in \mathrm{slice}(l)$ that isn't (properly) dominated by any $n' \in \mathrm{slice}(l)$ is called the *slice root* of l.

Example of Slices



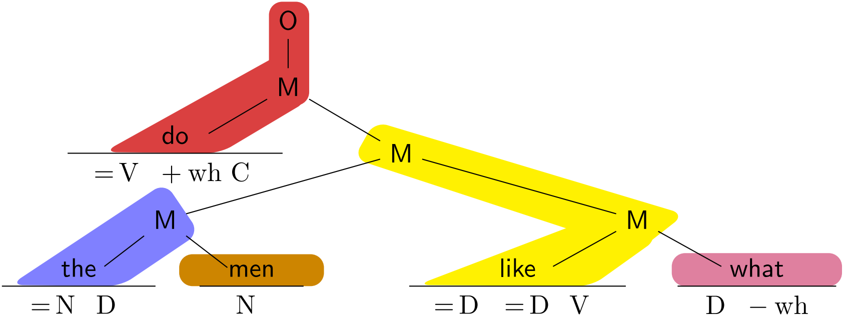
Simple Facts About Slices

- Every node of a derivation tree belongs to some slice.
- Slices are continuous.
- Moving from $\mathrm{slice}(l)$ to $\mathrm{slice}(l')$ such that l' was selected by l, one eventually reaches a slice of size 1.

Category Refinement Strategy

- Assume we are given an MG G and deterministic bottom-up automaton A.
- Subscript interior node labels with state of automaton, following Thatcher's strategy.
- Move subscript from slice root of lexical item to its category feature.
- Refine selection features accordingly.
- The set of final categories of the new MG G' is $\{c_q \mid c \in F_G, q \in F_A\}$.
- Note that only finitely many combinations of slices and states need to be considered, so the procedure can be carried out efficiently.

Two Examples of Category Refinement



MG G' for $G \cap ODD$

a :: $\mathrm{a_o}$

a :: $\mathrm{a_o} \; -\mathrm{k}$

b :: $=\mathrm{a_o} \; =\mathrm{a_o} \; +\mathrm{k} \; \mathrm{a_e}$

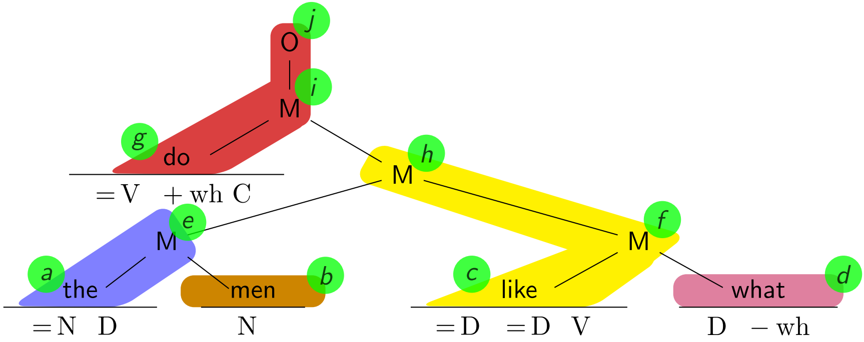
b :: $=\mathrm{a_o} \; =\mathrm{a_e} \; +\mathrm{k} \; \mathrm{a_o}$

b :: $=\mathrm{a_e} \; =\mathrm{a_o} \; +\mathrm{k} \; \mathrm{a_o}$

b :: $=\mathrm{a_e} \; =\mathrm{a_e} \; +\mathrm{k} \; \mathrm{a_e}$

c :: $=\mathrm{a_o} \; \mathrm{c_o}$

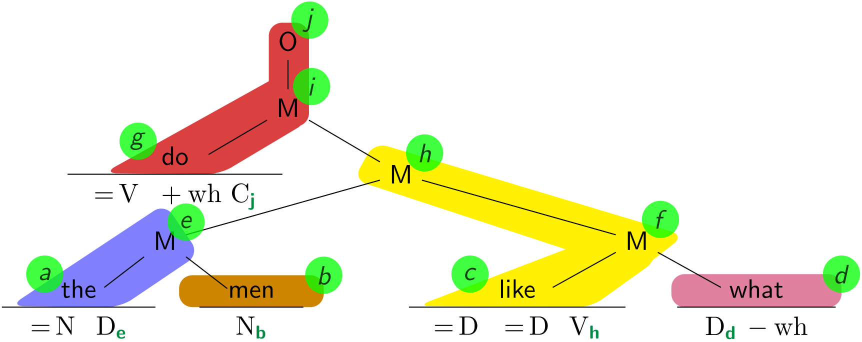
Two Examples of Category Refinement



MG G' for $G \cap ODD$

a :: a_o	b :: $= a_o \ = a_o + k \ a_e$	c :: $= a_o \ c_o$
a :: $a_o \ - k$	b :: $= a_o \ = a_e + k \ a_o$	
	b :: $= a_e \ = a_o + k \ a_o$	
	b :: $= a_e \ = a_e + k \ a_e$	

Two Examples of Category Refinement



MG G' for $G \cap ODD$

$\text{a} :: \mathrm{a_o}$

$\text{a} :: \mathrm{a_o} \; -\mathrm{k}$

$\text{b} :: \; =\mathrm{a_o} \; =\mathrm{a_o} \; +\mathrm{k} \; \mathrm{a_e}$

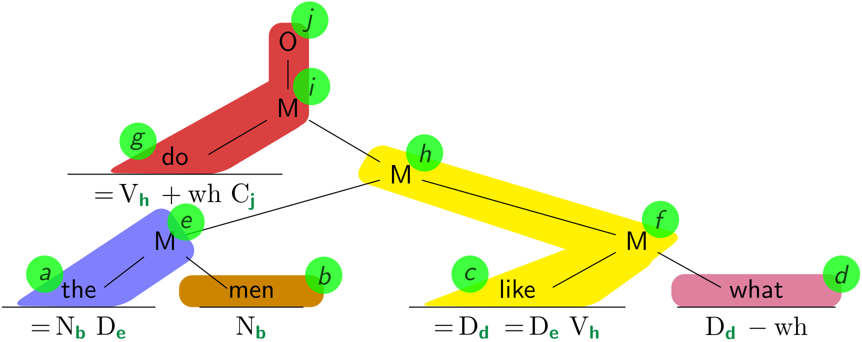
$\text{b} :: \; =\mathrm{a_o} \; =\mathrm{a_e} \; +\mathrm{k} \; \mathrm{a_o}$

$\text{b} :: \; =\mathrm{a_e} \; =\mathrm{a_o} \; +\mathrm{k} \; \mathrm{a_o}$

$\text{b} :: \; =\mathrm{a_e} \; =\mathrm{a_e} \; +\mathrm{k} \; \mathrm{a_e}$

$\text{c} :: \; =\mathrm{a_o} \; \mathrm{c_o}$

Two Examples of Category Refinement



MG G' for $G \cap ODD$

$\mathrm{a} :: \mathrm{a}_o$

$\mathrm{a} :: \mathrm{a}_o\ -\mathrm{k}$

$\mathrm{b} :: =\mathrm{a}_o\ =\mathrm{a}_o\ +\mathrm{k}\ \mathrm{a}_e$

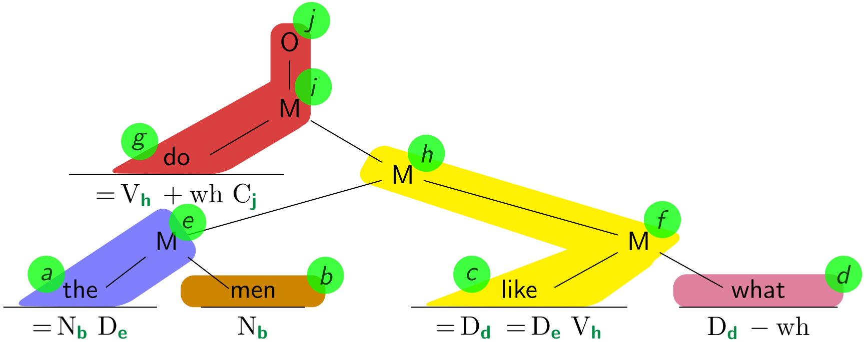
$\mathrm{b} :: =\mathrm{a}_o\ =\mathrm{a}_e\ +\mathrm{k}\ \mathrm{a}_o$

$\mathrm{b} :: =\mathrm{a}_e\ =\mathrm{a}_o\ +\mathrm{k}\ \mathrm{a}_o$

$\mathrm{b} :: =\mathrm{a}_e\ =\mathrm{a}_e\ +\mathrm{k}\ \mathrm{a}_e$

$\mathrm{c} :: =\mathrm{a}_o\ \mathrm{c}_o$

Two Examples of Category Refinement



MG G' for $G \cap ODD$

a :: a_o

a :: $a_o \ - k$

b :: $= a_o \ = a_o + k \ a_e$

b :: $= a_o \ = a_e + k \ a_o$

b :: $= a_e \ = a_o + k \ a_o$

b :: $= a_e \ = a_e + k \ a_e$

c :: $= a_o \ c_o$

Correctness of Procedure

- Suppose that $mder(G') \neq mder(G) \cap L(A)$.
- Then there must be some tree t such that $t \in mder(G')$ iff $\pi(t) \notin mder(G) \cap L(A)$. So $head(t)$ has a category feature c_q, but A does not assign state q to the root of $\pi(t)$.
- Since A is deterministic, such a situation may arise only if A entered the slice in a state that differs from the subscripts on the corresponding selector feature of $head(t)$.
- By induction on slices, we eventually reach a slice of size 1 to which A assigned a state that differs from the subscript of the category feature of its lexical item. But A is deterministic. Contradiction.

Further P-Closure Properties

P-Closure Corollaries

- The class of **MDTLs over Σ, *Feat*** is p-closed under
  - intersection,
  - relative complement.
- Given lexicon *Lex*, the class of **MDTLs over subsets of *Lex*** is p-closed under
  - complement,
  - union.
- For every regular tree language L and linear transduction τ with an MDTL as its co-domain, it holds that $\tau(L)$ is a projection of some MDTL.

Minimalist Grammars with Regular Control

Minimalist Grammars with Regular Control (MGRCs)

An MG is a 6-tuple $G := \langle \Sigma, Feat, F, Lex, Op, \mathcal{R} \rangle$, where

- Σ, *Feat* , F, *Lex*, and *Op* are defined as usual, and
- $\mathcal{R}$ is a finite collection of regular tree languages.

Its *controlled derivation tree language* is $cder(G) := mder(G) \cap \mathcal{R}$. The derived tree language of G (and its string yield) are obtained from $cder(G)$ via the mbutt of Kobele et al. (2007).

- MGRCs are **more succinct than their refined equivalent**.
- Given a lexicon *Lex* and $n \geq 0$, let $Lex^{(n)} := \{l \in Lex \mid l$ has exactly n selector features$\}$. In the worst case

$$|Lex_{G'}| = \sum_{i \geq 0} \left(|Lex_G^{(i)}| \cdot |Q|^{i+1} \right)$$

Minimalist Grammars with Regular Control

Minimalist Grammars with Regular Control (MGRCs)

An MG is a 6-tuple $G := \langle \Sigma, Feat, F, Lex, Op, \mathcal{R} \rangle$, where

- Σ, $Feat$, F, Lex, and Op are defined as usual, and
- $\mathcal{R}$ is a finite collection of regular tree languages.

Its *controlled derivation tree language* is $cder(G) := mder(G) \cap \mathcal{R}$. The derived tree language of G (and its string yield) are obtained from $cder(G)$ via the mbutt of Kobele et al. (2007).

- MGRCs are **more succinct than their refined equivalent**.
- Given a lexicon Lex and $n \geq 0$, let $Lex^{(n)} := \{l \in Lex \mid l$ has exactly n selector features$\}$. In the worst case

$$|Lex_{G'}| = \sum_{i \geq 0} \left(|Lex_G^{(i)}| \cdot |Q|^{i+1} \right)$$

Application 1: Reference-Set Computation

- Reference-set constraints are **economy conditions** similar to OT: Given some input tree t
  - compute the set of competing output candidates,
  - rank them according to some economy metric,
  - discard all sub-optimal candidates.
- Graf (2010a,b): Most reference-set constraints in the syntactic literature can be modelled by linear tree transductions. In particular, those **constraints act as filters**, so the transductions have MDTLs as their domain and co-domain.
- From the previous corollary for linear transductions it follows that **the expressivity of MGs is not increased**.

Application 2: Non-Local Dependencies without Movement

- Expletive constructions in English show subject-verb agreement even though no movement seems to be involved.

(1) a. There seem**s** to be **a man** in the garden.
 b. There seem to be **three men** in the garden.

- The subject position is filled by the expletive
$\Rightarrow$ arguably no movement

Proposal

Regular constraint operating on "pseudo-features" (not part of the MG itself) $\Rightarrow$ enforce non-local dependencies without movement

- Quite generally, this allows us to **enrich MGs with AGREE** (Chomsky 2000).

Application 3: Relativized Minimality

- In Minimalist syntax, the contrast below is explained by *Relativized Minimality*: If a movement licensor feature can be checked by two different phrases, **the closer one moves**.

(2) Who/what bought *t* who/what?

(3) * Who/What bought who/what *t*?

- Relativized Minimality relies on both *who* and *what* carrying a $-\mathrm{wh}$ feature. This idea conflicts with the SMC, and in order to derive (2), we must allow *who*/*what* to appear without a $-\mathrm{wh}$-feature. But then nothing in the MG blocks (3).

Proposal

Moving phrase XP with feature $-f$ to ZP is banned if there is a closer YP with pseudo feature $-f$.

Further Applications

- Island constraints
- Phases
- *that*-trace filter
- L-marking
- Limited feature percolation/Pied-Piping
- Control/Binding(?)

Conclusion

- MDTLs are not closed under intersection with regular tree languages.
- However, they enjoy **p-closure properties akin to regular languages**:
  - intersection,
  - intersection with regular tree languages,
  - union,
  - (relative) complement,
  - certain linear transductions.
- Hence, enriching MGs with regular control over their derivations does not increase their generative capacity.
- Numerous applications; in particular, ideas from model-theoretic syntax can be easily incorporated

References

Chomsky, Noam. 2000. Minimalist inquiries: The framework. In *Step by step: Essays on minimalist syntax in honor of Howard Lasnik*, ed. Roger Martin, David Michaels, and Juan Uriagereka, 89–156. Cambridge, Mass.: MIT Press.

Graf, Thomas. 2010a. Reference-set constraints as linear tree transductions via controlled optimality systems. In *Proceedings of the 15th Conference on Formal Grammar*. To appear.

Graf, Thomas. 2010b. A tree transducer model of reference-set computation. *UCLA Working Papers in Linguistics* 15:1–53.

Harkema, Henk. 2001. A characterization of minimalist languages. In *Logical aspects of computational linguistics (lacl'01)*, ed. Philippe de Groote, Glyn Morrill, and Christian Retoré, volume 2099 of *Lecture Notes in Artificial Intelligence*, 193–211. Berlin: Springer.

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

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

Rogers, James. 1998. *A descriptive approach to language-theoretic complexity*. Stanford: CSLI.

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.