Tree Adjunction as Minimalist Lowering

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

University of California, Los Angeles

tag+ 2012
September 27, 2012
MGs vs TAG

- **String Languages**
  \[ \text{CFG} \subseteq \text{LIG} \equiv \text{TAG} \equiv \text{CCG} \subseteq \text{LCFRS} \equiv \text{MCTAG} \equiv \text{MG} \]

- **Tree Languages**
  \[ \text{TAG} \not\subseteq \text{MG} \& \text{MG} \not\subseteq \text{TAG} \]

**Question**
Can MGs be extended to subsume TAG on a tree level?
Outline

1 Minimalist Grammars with Reset Lowering
   - Slices and Merge
   - Move & Reset Lowering

2 Translation from MGs to TAG
   - General Idea and Prerequisites
   - Initial Trees & Substitution
   - Tree Adjunction
   - Advanced Topics
Movement-Generalized MGs

**Standard MGs** (Stabler 1997, 2011)
- Inspired by Chomsky’s Minimalist Program
- Two structure building operations:
  - Merge (combines trees) and Move (displaces subtrees)
- Both operations are controlled by features on the lexical items.

**Movement-Generalized MGs** (Graf 2012)
- Extend MGs with a template for defining new variants of Move
  *without increasing weak generative capacity*
- Parameters: size of displaced constituent, linear order, direction of Move (upwards/downwards)
- Defined in terms of their (regular) derivation tree language plus a transduction to derived trees.
Defining MGs via Their Derivations: Slices

We start with a derivation-tree based definition of MGs without movement.

**Slices (≈ elementary trees/phrase projected by a lexical item)**

A *slice* is a strictly binary branching tree such that

- every interior node is labeled with a positive polarity Merge feature,
- every interior node is a mother of exactly one node labeled □,
- exactly one leaf node is a lexical item (the **head**) with a negative polarity Merge feature.

A Minimalist derivation is a combination of slices satisfying certain conditions.
Example: Slices and a Combination Thereof

\[
\begin{align*}
N^+ & \quad D^- \quad \square \\
\text{the} & \quad \square \\
D^+ & \quad \square \\
\text{kicked} & \quad V^- \quad \square \\
\end{align*}
\]

\[
\begin{align*}
D^+ & \quad \square \\
N^+ & \quad D^- \\
\text{the} & \quad \square \\
D^+ & \quad \square \\
N^+ & \quad D^- \\
\text{the} & \quad \square \\
N^+ & \quad D^- \\
\text{the} & \quad \square \\
N^+ & \quad D^- \\
\end{align*}
\]

\[
\begin{align*}
moose & \quad N^- \\
N^+ & \quad D^- \\
\text{the} & \quad \square \\
D^+ & \quad \square \\
\text{moose} & \quad \square \\
\end{align*}
\]
Conditions on Merge

**Constraint 1: Merge**

Every interior node with a positive polarity Merge feature $F^+$ immediately dominates the root of a slice whose head has the matching feature $F^-$. 

**Constraint 2: Final**

The head of the root of the derivation must have a distinguished `final` Merge feature.
Mapping to Derived Trees

Replace interior node labels by arrows pointing in the direction of the head of the slice.

Example

```
D^+  
/   \  
N^+   D^+  
/   /   \   \  
the D^- moose N^- kicked V^- N^+  
/   /   \   \   \  
the D^- moose N^-  
```
Replace interior node labels by arrows pointing in the direction of the head of the slice.

Example

```
>  
<  the :: D^-  
<  moose :: N^-  
<  kicked :: V^-  
<  the :: D^-  
<  moose :: N^-  
```
In derivation trees, Move is only indicated by unary branching — no actual displacement occurs before the mapping to derived trees.
In derivation trees, Move is only indicated by unary branching — no actual displacement occurs before the mapping to derived trees.
Slices Again

Slices Addendum

- A slice may contain unary branching nodes.
- All unary branching nodes — and only those — are labeled with a positive polarity Move feature with directionality $d \in \{\lambda, \rho\}$.
- A head’s negative polarity Merge feature may be followed by a finite number of negative Move features.
- Every Move feature furthermore has a non-negative size value indicating the root of the subtree to be displaced.
Example: Slices involving Move

\[ \text{which ::} \quad D^- \quad \text{wh}^-[1] \quad \square \]

\[ N^+ \]

\[ T^+ \]

\[ t^+_\rho[0] \quad \square \]

\[ \varepsilon :: C^- \]

\[ X^+ \]

\[ \text{wh}^+\lambda[5] \quad \square \]

\[ \text{top}^+_\rho[3] \quad \square \]

\[ Y^+ \]

\[ \varepsilon :: Z^- \quad \text{wh}^- \quad \square \]
What are the Relevant Move Nodes?

Finding Occurrences for Reset Lowering

Move node $m$ with feature $f^+[i]$, $i \geq 0$, is an occurrence of head $h$ iff

- $h$ has a matching feature $f^- [i]$, and
- the $i$-th node $n$ of the slice of $h$ c-commands $m$ in the derivation tree, and
- there is no head $h'$ satisfying the previous conditions that is c-commanded by $n$. 
Find the Occurrences!

\[ D^+ \]

\[ d :: D^- \]

\[ Z^+ \]

\[ X^+ \]

\[ x :: X^- \]

\[ Z^+ \]

\[ f^+ [1] \]

\[ x :: X^- \]

\[ f^+ [1] \]

\[ z :: Z^- f^- [1] \]

\[ z :: Z^- \]
Find the Occurrences!
Constraints on Move

**Constraint 1: Move**

For every head $h$ with $n$ negative Move features, $n \geq 1$, there exist $n$ distinct Move nodes that are occurrences of $h$.

**Constraint 2: SMC**

Every Move node is an occurrence of exactly one head.

**Corollary for Reset Lowering**

- No head has two negative Move features with both identical feature names and identical size values.
- The order of a head’s negative Move features is irrelevant.
Constraints on Move

Constraint 1: Move
For every head $h$ with $n$ negative Move features, $n \geq 1$, there exist $n$ distinct Move nodes that are occurrences of $h$.

Constraint 2: SMC
Every Move node is an occurrence of exactly one head.

Corollary for Reset Lowering
- No head has two negative Move features with both identical feature names and identical size values.
- The order of a head’s negative Move features is irrelevant.
General Strategy

- **Given:** derivation tree language of some TAG $G$
- **Step 1:** Put $G$ into a particular normal form.
- **Step 2:** Define a mapping from TAG derivations to Minimalist derivations.
  - Adjunction is Merger of auxiliary tree $T$ at adjunction site $A$ followed by lowering of the material below $A$ to $T$’s foot node.
- **Step 3:** Ensure the output is an MDTL.
**Definition (TAG Derivation Tree)**

A **TAG derivation tree** is a finite tree with each node’s label consisting of:

- the **name** of an elementary tree $e$, and
- the **address** of the node where $e$ is adjoined/substituted (if such a node exists).

**Example**

```
A: S^ε
  NP^0  VP^1
    V^10   NP^11

B: VP^ε
  Aux^0  VP^1

A, B,1
```
A **TAG derivation tree** is a finite tree with each node’s label consisting of

- the **name** of an elementary tree $e$, and
- the **address** of the node where $e$ is adjoined/substituted (if such a node exists).

**Example**

```
A: S^ε
  NP^0  VP^1
     |     |
    V^10  NP^11

B: VP^ε
  Aux^0  VP^1
     |     |
    B,1
```
All elementary trees must be

- strictly binary branching, and
- projective.

**Definition (Projectivity)**

Every interior node is a projection of some (possibly empty) leaf that is neither a foot node nor a substitution node.
Initial Trees

Trees containing neither foot nodes nor substitution nodes are straight-forward, thanks to projectivity:

Example

```
ZP
  /
 y  z
```

```
Y^+
  /
 y ::  Y^-  z ::  Z^-
```
Substitution is handled by Merge, too:

Example

```
  ZP
 /   \
|     |
DP↓   ZP
 /     /
|     |  
y   z

DP
 /   \
|     |
d   n

D^+  N^+  Y^+
d :: D^-  n :: N^-  y :: Y^-  z :: Z^-
```
Tree Adjunction

Tree Adjunction ≡ Merge + Reset Lowering
Comparing the Derived Trees

ZP

DP

ZP

d n x ZP

<

d n >

ε

x <

z >

y z
An Example with Multiple Adjunctions
An Example with Multiple Adjunctions

Observation

An elementary tree may have multiple MG correspondents.
An Example with Multiple Adjunctions

Z⁺

X⁺

f⁺[3]

N⁺

D⁺

z :: Z⁻

d :: D⁻

n :: N⁻

Z⁺

X⁺

x :: X⁻

f⁺[1]

Y⁺

z :: Z⁻

y :: Y⁻

z :: Z⁻

Observation

An elementary tree may have multiple MG correspondents.
Another Example with Multiple Adjunctions
Another Example with Multiple Adjunctions

Observation
A single feature name suffices for all instances of reset lowering.
Another Example with Multiple Adjunctions

A single feature name suffices for all instances of reset lowering.
But is it a Minimalist Derivation Tree Language?

- The output $L$ of the translation might not be a well-formed MDTL (some combinations of slices might be missing).

- However:
  - TAG derivation tree languages are regular,
  - the translation is a linear tree transduction,
  - regular tree languages are closed under linear tree transduction,
  - MDTLs are (almost) closed under intersection with regular tree languages (Graf 2011; Kobele 2011).

- Take the smallest superset $L'$ of $L$ that is an MDTL ($L'$ is guaranteed to exist) and intersect it with $L$.

- This yields the MDTL of some MG that generates all derived trees of the original TAG, and only those.
Expressivity of MGs with Reset Lowering

- Even with only one feature name for reset lowering it is still possible to generate
  
  $a_1^n \cdot a_2^n \cdots a_{k-1}^n \cdot a_k^n$

  for any $k \geq 1$.

- This is so because features are considered identical by the SMC only if they have the same size value.
  
  $\Rightarrow$ size value can emulate additional feature names

- If the SMC ignores the size value, only TALs can be generated.
Conclusion

- **Issue**
  - MGs have greater weak generative capacity than TAG.
  - Still the two generate incomparable classes of tree languages.
  - Can this gap be bridged?

- **Solution**
  - Adjunction cuts a tree $t$ into two halves $t_1$ and $t_2$, inserts new material and puts it all back together.
  - MGs generate the auxiliary tree in the intended position and lower $t_2$ to the foot node.

- **Future Research**
  - does not generalize well to higher-order TAG (Rogers 2003)
    - MGs with multiple feature names resemble MCTAG
  - Reset Lowering is not a particularly natural movement type.
  - Sideward Movement should also work, though.
  - More generally: What property must a movement type satisfy in order to subsume (higher-order) Tree Adjunction?
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.

Graf, Thomas. 2012. Movement-generalized minimalist grammars. In *LACL 2012*, ed. Denis Béchet and Alexander J. Dikovsky, volume 7351 of *Lecture Notes in Computer Science*, 58–73.

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.

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

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.