Derivational Minimalism

Stabler (1997)

MATTHEW KUHN

GRAMMARS FOR TREES AND GRAPHS

SAARLAND UNIVERSITY 2018
Derivational Minimalism

I. Outline
 II. Motivation
   A. Agreement
   B. Dependencies
   C. Generalization
   D. Rules

III. Minimalism
   B. Derivational Minimalism - Stabler (1997)

IV. How it Works
   A. Core Concepts
   B. Merge
   C. Features
   D. Move

V. Formal Definition
   A. Definition of Minimalist Grammar
   B. Maximal Projection
   C. Specifiers vs. Complements

VI. Putting it all Together
VII. Questions
Motivation

● Problems with grammars we’ve seen
1. Agreement
2. Dependencies
3. Generalization
4. Rules
Motivation

1. Agreement
   ● One solution: Separate symbols
   ● Problem: Fails to capture intuitive generalizations
     ○ e.g. “leave” and “leaves” are inflected forms of same verb
Motivation

2. Dependencies
   ● Problem: Mechanically difficult, unintuitive
     ○ very language-specific
     ○ requires indexing of gaps and fillers
   ● Easier in TAGs than PSGs, but same problems exist
Motivation

3. Generalization
   - Separate rules for each language
   - Problem: this contradicts Chomskyan prediction of Universal Grammar
Motivation

4. Rules
   - PSGs
     ○ require large numbers of PS rules in addition to the lexicon
   - TAGs
     ○ require large numbers of trees in addition to lexicon
   - Intuitively, lexemes select their rules/complements
   - Can we model this in a single structure?

I. Outline
   II. Motivation
      - Agreement
      - Dependencies
      - Generalization
      - Rules
   III. Minimalism
      - Derivational Minimalism - Stabler (1997)
   IV. How it Works
      - Core Concepts
      - Merge
      - Features
      - Move
   V. Formal Definition
      - Definition of Minimalist Grammar
      - Maximal Projection
      - Specifiers vs. Complements
   VI. Putting it all Together
   VII. Questions
Minimalism

The Minimalist Program - Chomsky (1995)

Goals:

● Perfection
  ○ Principles fixed for all languages, with a set of binary switches to modulate characteristics

● Economy of Derivation
  ○ Transformations happen only to match interpretable features (e.g. English plural inflection) with uninterpretable ones (e.g. English verb agreement)

● Economy of Representation
  ○ Structures only as complex as necessary

Downside: Not very formal
Derivational Minimalism - Stabler (1997)

Goals

- Rigorously formalize the ideas behind the Minimalist Program
- Provide a framework that can be used to represent most, if not all, languages
How It Works

Core Concepts

- Represent lexical items with feature structures
- Two primary operations: Merge & Move
  - In both cases: Combine two items when their features match
- Algebraic cancellation
  - cancel similar labels from left to right until only start symbol remains
How It Works - Merge

- Combines elements from the lexicon or tree
- Move is basically just a special kind of merge

Ex: creating the VP “makes tortillas”

Items:
=\{d \equiv d \, v \, make\}
\{d \, tortillas\}

Diagram:
```
<
\equiv d \, v \, make
\equiv d \, tortillas
```
How It Works - Features

aka “What the heck do all these letters mean?”

- Two types of features
  1. Bases and selectors
     Think of these as “merge features”
  2. Licensors and Licensees
     These are “move features”

- Features have polarity: a positive feature plus a negative feature cancels and takes an action
Merge Features

- Bases are traditional lexical categories
  - ex: D, N, P, V
- Selectors trigger cancellation
  - Can be either weak or strong
  - Weak: =d, =n
    Just cancel
  - Strong: =D, D=, =N, N=
    Cancel and moves phonetic content of the base
    Think of the = like a plug telling you where to attach
Example:
=d =d v make
d tortillas
How It Works - Features

Move Features

● Arbitrary names
  ○ Usually indicates a function
  ○ ex: +case, -wh

●Licensors are positive, stationary, can be strong/weak
  ○ Weak: +case, +wh
    only semantic content moves (e.g. for scope)
  ○ Strong: +CASE, +WH
    semantic and phonetic content moves

● Licensees are negative, indicate what moves
  ○ ex: -case, -wh
How It Works - Move

Example: “Maria will make tortillas”

Moving leaves a trace behind, but since the full content moved, no index is needed - it’s truly null
A minimalist grammar is a 4-tuple \((V, \text{Cat}, \text{Lex}, \mathcal{F})\), where

\[
V = (P \cup I), \\
\text{Cat} = (\text{base} \cup \text{select} \cup \text{licensee} \cup \text{licensor}), \\
\text{Lex} \text{ is a finite set of expressions built from } V \text{ and } \text{Cat} \\
\text{(where an “expression” is a tree as defined above), and} \\
\mathcal{F} = \{\text{merge, move}\}.
\]

P = Phonetic features  
I = Interpreted (semantic) Features  
Expression = \(V + \text{Cat}\) for each lexeme (thought of as trees)  
Order: select > licensor > base > licensee > P > I  
Note: Not all of these must be defined - MG allows for null P & I as well as elements from Cat  
All trees are binary
Formal Definition

Two Important Concepts

Maximal Projection

- When nodes combine, one “projects” over the other
- Quick and dirty: projection ~ head relationship
- Indicated with > and < in our trees (points towards head)
- Maximal Projection: the node y closest to the root that still has X as its head
  - Alternately, it’s the nodes and edges that include x and y
- Why do we care?
  - Movement moves the entire maximal projection of the -x head to be the specifier of the +x/+X node
Formal Definition

Two Important Concepts

Specifiers vs. Complements
- Specifiers precede their head
- Complements are preceded by their head
Tlhingan Hol boghojmeH QaQ jajvam!
Today is a good day to learn Klingon!
Putting it All Together

A bit about Klingon
- Created by Marc Okrand in 1984 for Star Trek III: The Search for Spock
- Unusual phonetic inventory
  - it lacks /a/ but has /ɑ/, unusual consonants like /tl/
- No adjectives
  - uses verbs with meanings like “to be big” instead
- Agglutinative
  - adds affixes to modify meaning of words
- OVS word order
  - Less than 35000 humans speak a language like this natively

The word order gives us the opportunity to see how MG handles languages very unlike English.
Putting it All Together

Sample sentence:
Hoch jaghpu’ HoH ‘op Suviw’
every enemy kills some warrior
translation: “Some warrior kills every enemy”

Lexicon:
=nd -q ‘op (some)
=nd -case Hoch (every)
n Suviw’ (warrior)
n jaghpu’ (enemy)
=d =d v HoH (kills)
=ť +q c ε
=V +CASE ť ε
Putting it All Together

Step 1) Merge

=₇ d-case Hoch (every) + ṃ jaghpu' (enemy)
Putting it All Together

Step 2) Merge

\[ =d =d \land HoH \text{ (kills)} + \]

---

I. Outline

II. Motivation
   A. Agreement
   B. Dependencies
   C. Generalization
   D. Rules

III. Minimalism
   A. The Minimalist Program
   B. Derivational Minimalism

IV. How it Works
   A. Core Concepts
   B. Merge
   C. Features
   D. Move

V. Formal Definition
   A. Definition of Minimalist Grammar
   B. Maximal Projection
   C. Specifiers vs. Complements

VI. Putting it All Together

VII. Questions
Putting it All Together

Step 3) Merge

\[ \Rightarrow d -q 'op (some) + Suvìw' (warrior) \]
Putting it All Together

Step 4) Merge
Putting it All Together

Step 5) Merge with head movement

\[ = V + \text{CASE} \uparrow \varepsilon + \]
Putting it All Together

Step 6) Overt Move
Putting it All Together

Step 7) Merge

\[ \hat{\epsilon} + q \times \epsilon \]
Putting it All Together

Step 8) Covert Move

Qapla! (Success!)

Outline

Motivation
A. Agreement
B. Dependencies
C. Generalization
D. Rules

Minimalism
A. The Minimalist Program
B. Derivational Minimalism

How it Works
A. Core Concepts
B. Merge
C. Features
D. Move

Formal Definition
A. Definition of Minimalist Grammar
B. Maximal Projection
C. Specifiers vs. Complements

Putting it All Together

Questions
Questions / References
