Deconstructing Syntactic Generalizations with Minimalist Grammars

THE UNIVERSITY OF CHICAGO

Marina Ermolaeva
mail@mermolaeva.com

Introduction

- **Goal:** derive analyses of syntactic phenomena based on a quantitative metric and in a way compatible with theoretical literature
- Convert a naive grammar into a linguistically motivated one via interpretable steps
- Factor out syntactic generalizations and express them as new lexical items

Minimalist grammars

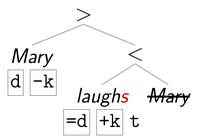
- Introduced by Stabler (1997)
- Formalize the Minimalist framework for syntax (Chomsky 1995)
- Lexical items (LIs): pairs consisting of a string and syntactic features
- Operations: **merge** and **move**, which consume matching features
- Complex words are formed via **merge** + head movement
- Syntactic features have a **name** (from some set *Base*) and a **type**:

	Attractor	Attractee
merge	=x (right) x= (left) =>x (head mvt)	x (category)
move	+x (overt) *x (covert)	-x (licensee)

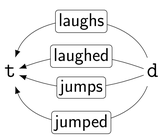
Grammar

Mary : : d -k
laughs : : =d +k t
laughed : : =d +k t
jumps : : =d +k t
jumped : : =d +k t

Derived tree



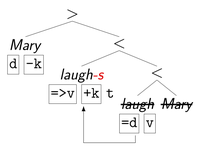
Multigraph

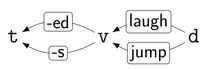


Metrics

|*Base*| 3
$\sum_{\text{syn}}$ 14
$\sum_{\text{phon}}$ 28
bits 317.78

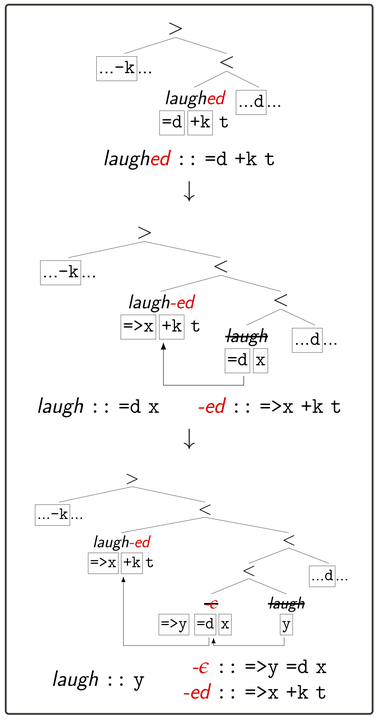
Mary : : d -k
-s : : =>v +k t
-ed : : =>v +k t
laugh : : =d v
jump : : =d v





|*Base*| 4
$\sum_{\text{syn}}$ 12
$\sum_{\text{phon}}$ 16
bits 236.16

Decomposing lexical items



- **LI decomposition** (Kobele 2018, to appear): split an LI into two and add a new feature to *Base*

$$w :: \alpha\beta x\gamma \quad \rightarrow \quad \begin{array}{l} u :: \alpha y \\ v :: {=}y\beta x\gamma \\ w = u \oplus v \end{array}$$

- For now: define $\oplus$ as concatenation

Operations

- **Batch decomposition:**

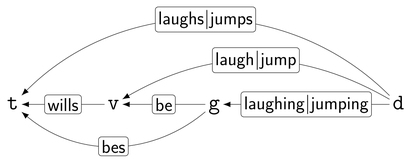
laugh : : =d v
laughing : : =d prog
laughs : : =d +k t
↓
laugh : : =d x
-ϵ : : =>x v
-ing : : =>x prog
-s : : =>x +k t

- **Contraction:**
remove an LI *-ϵ* : : =>x y; replace all instances of x and y with a new feature
- **Deletion:**
remove an LI *-ϵ* : : =>x y if the grammar provides an alternate path of empty LIs from x to y

Transforming a grammar

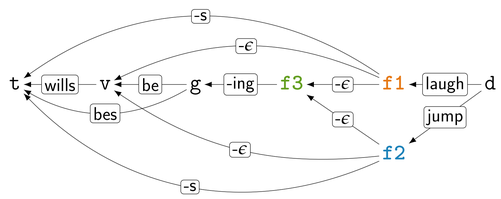
Original lexicon:

Mary : : d -k
bes : : =g +k t
wills : : =v +k t
be : : =g v
laughs : : =d +k t
laughing : : =d g
laugh : : =d v
jumps : : =d +k t
jumping : : =d g
jump : : =d v



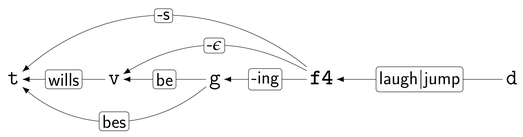
|*Base*| 5
$\sum_{\text{syn}}$ 24
$\sum_{\text{phon}}$ 49
bits 565.54

Decomposition steps: *laugh*, *jump*, *-ing*



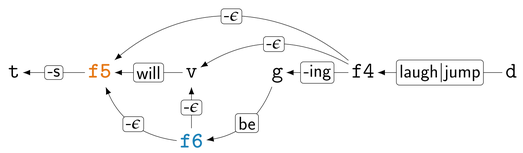
|*Base*| 8
$\sum_{\text{syn}}$ 30
$\sum_{\text{phon}}$ 28
bits 544.62

Contraction steps: *-ϵ* : : =>f1 f3, *-ϵ* : : =>f1 f2



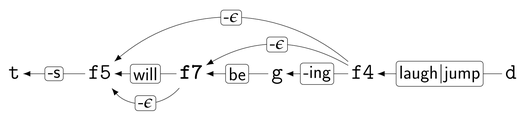
|*Base*| 6
$\sum_{\text{syn}}$ 21
$\sum_{\text{phon}}$ 27
bits 415.11

Decomposition steps: *-s*, *be*



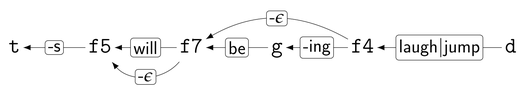
|*Base*| 8
$\sum_{\text{syn}}$ 23
$\sum_{\text{phon}}$ 23
bits 431.39

Contraction step: *-ϵ* : : =>f6 v



|*Base*| 7
$\sum_{\text{syn}}$ 21
$\sum_{\text{phon}}$ 23
bits 401.82

Deletion step: *-ϵ* : : =>f4 f5



|*Base*| 7
$\sum_{\text{syn}}$ 19
$\sum_{\text{phon}}$ 23
bits 375.03

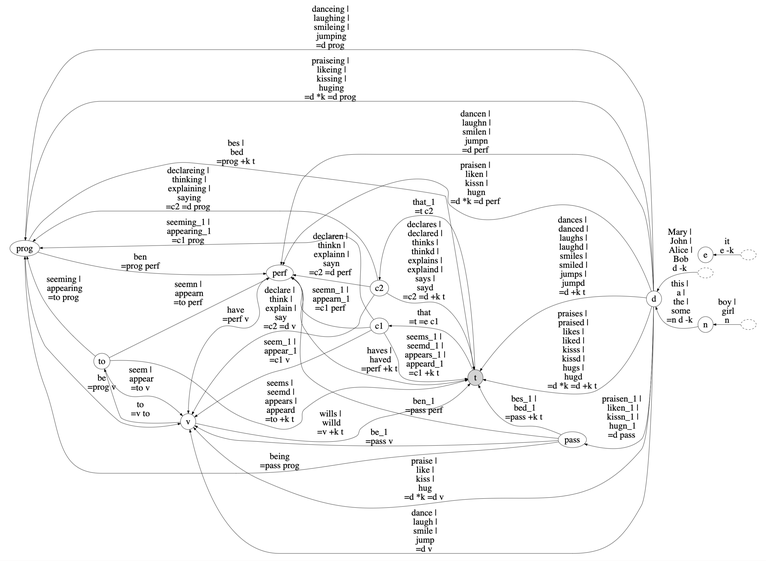
Final lexicon:

Mary : : d -k
laugh : : =d f4
jump : : =d f4
-s : : =>f5 +k t
-ϵ : : =>f7 f5
-ϵ : : =>f4 f7
will : : =f7 f5
be : : =g f7
-ing : : =>f4 g

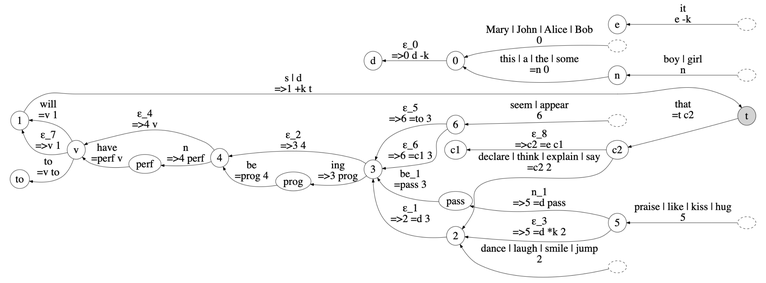
Automated decomposition

- A prototype Python implementation
- Input: naive minimalist grammar over unsegmented words
- Beam search to navigate the space of grammars defined by the operations
- Cost function: size in bits + heuristics

Input: 7572.80 bits



Output: 1964.68 bits



Examples: raising and expletive *it*