Capturing Online SRC/ORC Effort with Memory Measures from a Minimalist Parser

Aniello De Santo
he/him

aniellodesanto.github.io
aniello.desanto@utah.edu

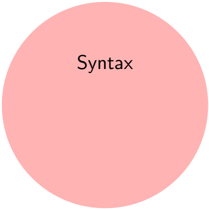
CMCL 2025

One Big Question

Which aspects of grammar influence sentence processing?

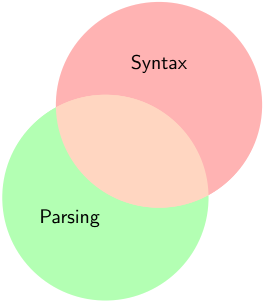
One Big Question

Which aspects of grammar influence sentence processing?



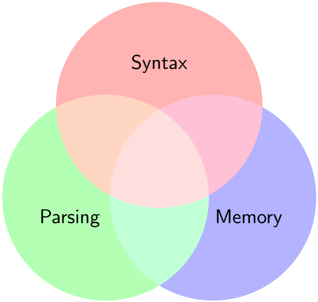
One Big Question

Which aspects of grammar influence sentence processing?



One Big Question

Which aspects of grammar influence sentence processing?



1. Do structure building operations predict behavioral results?
2. How do structure building/memory metrics fare wrt expectation based ones?

Forward to the Past

(How much) does grammatical structure matter in sentence processing?

A realistic grammar should [...] contribute to the explanation of linguistic behavior and to our larger understanding of the human faculty of language.

(Bresnan 1978: pg. 58)

Derivational Theory of Complexity (Miller and Chomsky, 1963)

- Processing complexity ∼ length of a derivation (Fodor & Garrett 1967; Berwick & Weinberg 1983)
- Essentially: there is a **cost** to mental computations.

- What is the right notion of syntactic derivation?
- What is costly? And why?

Forward to the Past

(How much) does grammatical structure matter in sentence processing?

A realistic grammar should [...] contribute to the explanation of linguistic behavior and to our larger understanding of the human faculty of language.

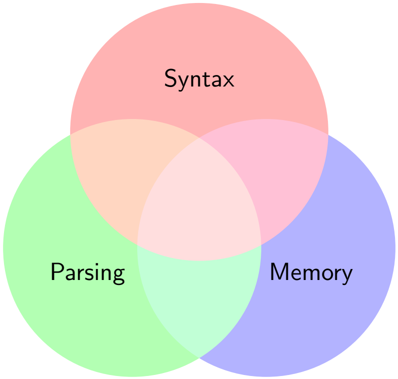
(Bresnan 1978: pg. 58)

Derivational Theory of Complexity (Miller and Chomsky, 1963)

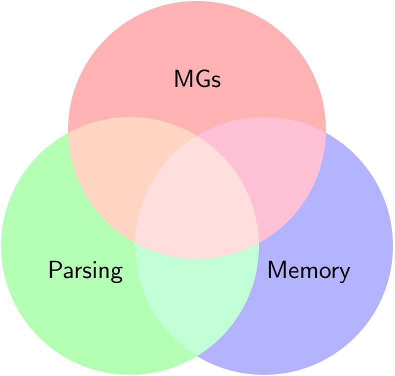
- Processing complexity $\sim$ length of a derivation (Fodor & Garrett 1967; Berwick & Weinberg 1983)
- Essentially: there is a **cost** to mental computations.

- What is the right notion of syntactic derivation?
- What is costly? And why?

A Formal Model of Sentence Processing

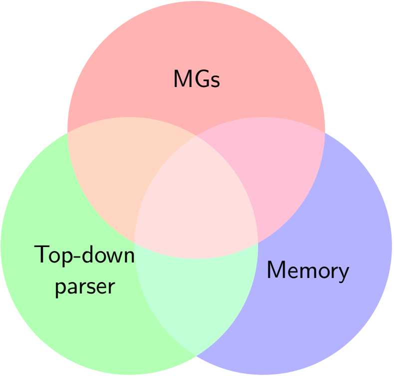


A Formal Model of Sentence Processing



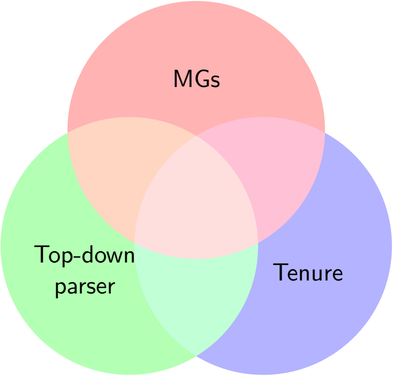
1. An explicit syntactic theory → Minimalist grammars (MGs)

A Formal Model of Sentence Processing



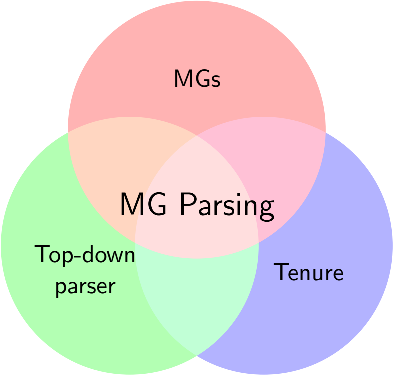
1. An explicit syntactic theory → Minimalist grammars (MGs)
2. A theory of how structures are built → top-down parser

A Formal Model of Sentence Processing



1. An explicit syntactic theory → Minimalist grammars (MGs)
2. A theory of how structures are built → top-down parser
3. A psychologically grounded linking theory → tenure

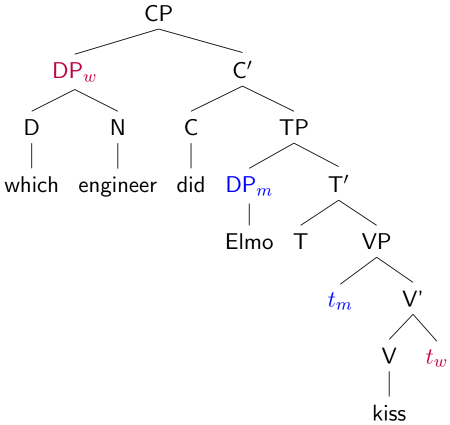
A Formal Model of Sentence Processing



1. An explicit syntactic theory → Minimalist grammars (MGs)
2. A theory of how structures are built → top-down parser
3. A psychologically grounded linking theory → tenure

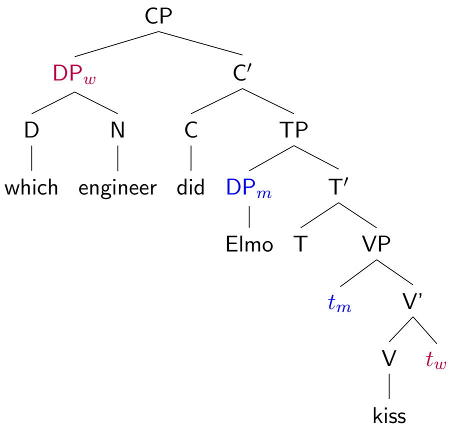
Outline

Minimalist Grammars (MGs) & Derivation Trees

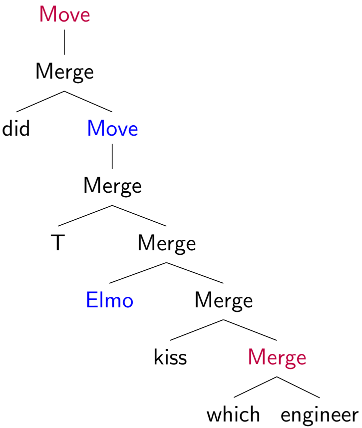


Phrase Structure Tree

Minimalist Grammars (MGs) & Derivation Trees

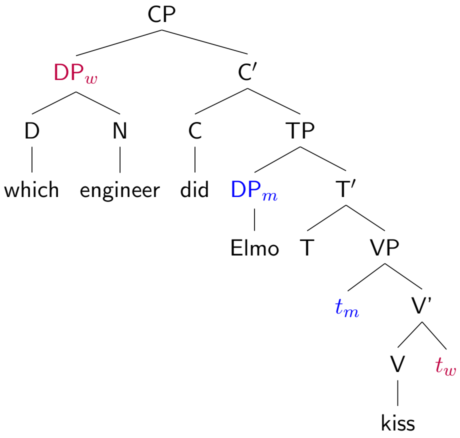


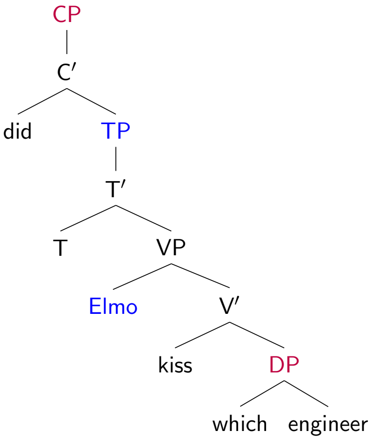
Phrase Structure Tree



Derivation Tree

MG Syntax: Derivation Trees



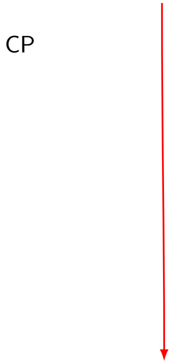


Phrase Structure Tree **Derivation Tree**

The Intuition: Top-Down MG Parsing

Who does Salem mock?

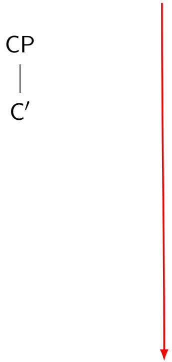
The Intuition: Top-Down MG Parsing



Who does Salem mock?

- Builds the structure from top to bottom
- Takes elements in an out of memory
- Complexity of the structure ≈ how much memory is used!

The Intuition: Top-Down MG Parsing

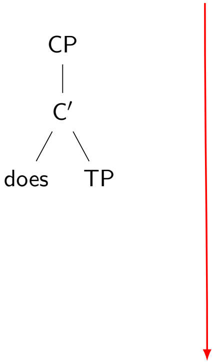


Who does Salem mock?

- Builds the structure from top to bottom
- Takes elements in an out of memory
- Complexity of the structure ≈ how much memory is used!

The Intuition: Top-Down MG Parsing

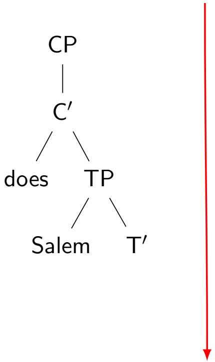
Who does Salem mock?



- Builds the structure from top to bottom
- Takes elements in an out of memory
- Complexity of the structure ≈ how much memory is used!

The Intuition: Top-Down MG Parsing

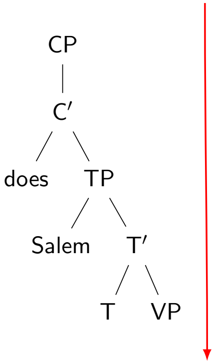
Who does Salem mock?



- Builds the structure from top to bottom
- Takes elements in an out of memory
- Complexity of the structure ≈ how much memory is used!

The Intuition: Top-Down MG Parsing

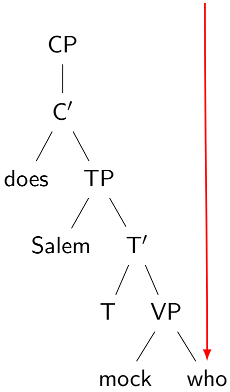
Who does Salem mock?



- Builds the structure from top to bottom
- Takes elements in an out of memory
- Complexity of the structure ≈ how much memory is used!

The Intuition: Top-Down MG Parsing

Who does Salem mock?



- Builds the structure from top to bottom
- Takes elements in an out of memory
- Complexity of the structure ≈ how much memory is used!

Incremental Top-Down Parsing

Technical details!

- String-driven recursive descent parser (Stabler 2013)

- • Who • does • Salem • T • mock

step 1 CP is conjectured
step 2 CP expands to C′
step 3 C′ expands to does and TP
step 4 TP expands to Salem and T′
step 5 T′ expands to T and VP
step 6 VP expands to mock and who
step 7 who is found
step 8 does is found
step 9 Salem is found
step 10 T is found
step 11 mock is found

Incremental Top-Down Parsing

Technical details!

- String-driven recursive descent parser (Stabler 2013)

[1]CP

- • Who • does • Salem • T • mock

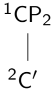
step 1	*CP* is conjectured
step 2	*CP* expands to *C'*
step 3	*C'* expands to *does* and *TP*
step 4	*TP* expands to *Salem* and *T'*
step 5	*T'* expands to *T* and *VP*
step 6	*VP* expands to *mock* and *who*
step 7	*who* is found
step 8	*does* is found
step 9	*Salem* is found
step 10	*T* is found
step 11	*mock* is found

Incremental Top-Down Parsing

Technical details!

- String-driven recursive descent parser (Stabler 2013)

- • Who • does • Salem • T • mock



step 1 *CP* is conjectured
step 2 *CP* expands to *C'*
step 3 *C'* expands to *does* and *TP*
step 4 *TP* expands to *Salem* and *T'*
step 5 *T'* expands to *T* and *VP*
step 6 *VP* expands to *mock* and *who*
step 7 *who* is found
step 8 *does* is found
step 9 *Salem* is found
step 10 *T* is found
step 11 *mock* is found

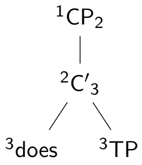
Incremental Top-Down Parsing

Technical details!

- String-driven recursive descent parser (Stabler 2013)

- • Who • does • Salem • T • mock

step 1 *CP* is conjectured
step 2 *CP* expands to *C'*
step 3 *C'* expands to *does* and *TP*
step 4 *TP* expands to *Salem* and *T'*
step 5 *T'* expands to *T* and *VP*
step 6 *VP* expands to *mock* and *who*
step 7 *who* is found
step 8 *does* is found
step 9 *Salem* is found
step 10 *T* is found
step 11 *mock* is found



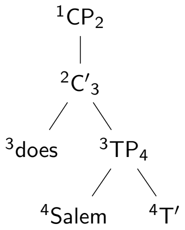
Incremental Top-Down Parsing

Technical details!

- String-driven recursive descent parser (Stabler 2013)

- • Who • does • Salem • T • mock

step 1	*CP* is conjectured
step 2	*CP* expands to *C'*
step 3	*C'* expands to *does* and *TP*
step 4	*TP* expands to *Salem* and *T'*
step 5	*T'* expands to *T* and *VP*
step 6	*VP* expands to *mock* and *who*
step 7	*who* is found
step 8	*does* is found
step 9	*Salem* is found
step 10	*T* is found
step 11	*mock* is found



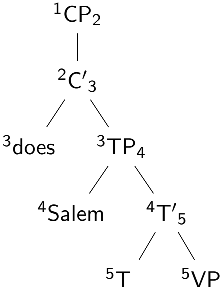
Incremental Top-Down Parsing

Technical details!

- String-driven recursive descent parser (Stabler 2013)

- • Who • does • Salem • T • mock

step 1 *CP* is conjectured
step 2 *CP* expands to *C'*
step 3 *C'* expands to *does* and *TP*
step 4 *TP* expands to *Salem* and *T'*
step 5 *T'* expands to *T* and *VP*
step 6 *VP* expands to *mock* and *who*
step 7 *who* is found
step 8 *does* is found
step 9 *Salem* is found
step 10 *T* is found
step 11 *mock* is found



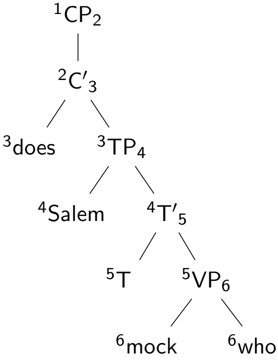
Incremental Top-Down Parsing

Technical details!

- String-driven recursive descent parser (Stabler 2013)

- • Who • does • Salem • T • mock

step 1	*CP* is conjectured
step 2	*CP* expands to *C'*
step 3	*C'* expands to *does* and *TP*
step 4	*TP* expands to *Salem* and *T'*
step 5	*T'* expands to *T* and *VP*
step 6	*VP* expands to *mock* and *who*
step 7	*who* is found
step 8	*does* is found
step 9	*Salem* is found
step 10	*T* is found
step 11	*mock* is found



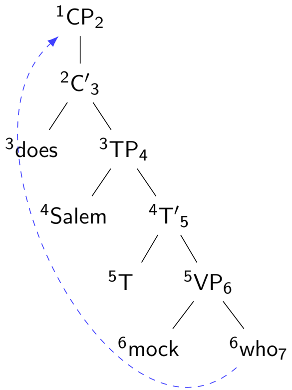
Incremental Top-Down Parsing

Technical details!

- String-driven recursive descent parser (Stabler 2013)

- • Who • does • Salem • T • mock

step 1 *CP* is conjectured
step 2 *CP* expands to *C'*
step 3 *C'* expands to *does* and *TP*
step 4 *TP* expands to *Salem* and *T'*
step 5 *T'* expands to *T* and *VP*
step 6 *VP* expands to *mock* and *who*
step 7 *who* is found
step 8 *does* is found
step 9 *Salem* is found
step 10 *T* is found
step 11 *mock* is found



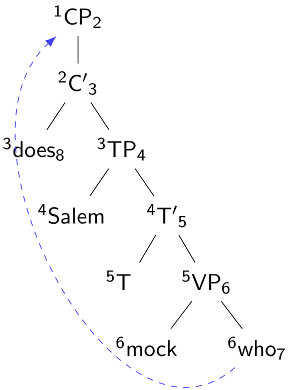
Incremental Top-Down Parsing

Technical details!

- String-driven recursive descent parser (Stabler 2013)

- • Who • does • Salem • T • mock

step 1	*CP* is conjectured
step 2	*CP* expands to *C'*
step 3	*C'* expands to *does* and *TP*
step 4	*TP* expands to *Salem* and *T'*
step 5	*T'* expands to *T* and *VP*
step 6	*VP* expands to *mock* and *who*
step 7	*who* is found
step 8	*does* is found
step 9	*Salem* is found
step 10	*T* is found
step 11	*mock* is found



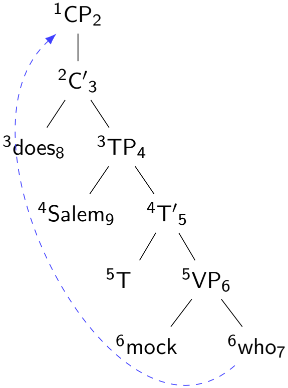
Incremental Top-Down Parsing

Technical details!

- String-driven recursive descent parser (Stabler 2013)

- • Who • does • Salem • T • mock

step 1	*CP* is conjectured
step 2	*CP* expands to *C'*
step 3	*C'* expands to *does* and *TP*
step 4	*TP* expands to *Salem* and *T'*
step 5	*T'* expands to *T* and *VP*
step 6	*VP* expands to *mock* and *who*
step 7	*who* is found
step 8	*does* is found
step 9	*Salem* is found
step 10	*T* is found
step 11	*mock* is found



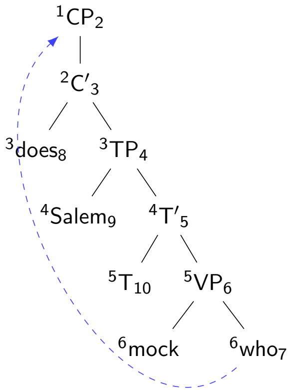
Incremental Top-Down Parsing

Technical details!

- String-driven recursive descent parser (Stabler 2013)

- • Who • does • Salem • T • mock

step 1	*CP* is conjectured
step 2	*CP* expands to *C'*
step 3	*C'* expands to *does* and *TP*
step 4	*TP* expands to *Salem* and *T'*
step 5	*T'* expands to *T* and *VP*
step 6	*VP* expands to *mock* and *who*
step 7	*who* is found
step 8	*does* is found
step 9	*Salem* is found
step 10	*T* is found
step 11	*mock* is found



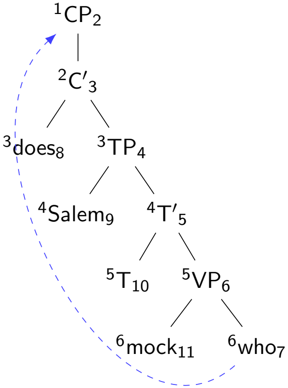
Incremental Top-Down Parsing

Technical details!

- String-driven recursive descent parser (Stabler 2013)

- • Who • does • Salem • T • mock

step 1	*CP* is conjectured
step 2	*CP* expands to *C'*
step 3	*C'* expands to *does* and *TP*
step 4	*TP* expands to *Salem* and *T'*
step 5	*T'* expands to *T* and *VP*
step 6	*VP* expands to *mock* and *who*
step 7	*who* is found
step 8	*does* is found
step 9	*Salem* is found
step 10	*T* is found
step 11	*mock* is found



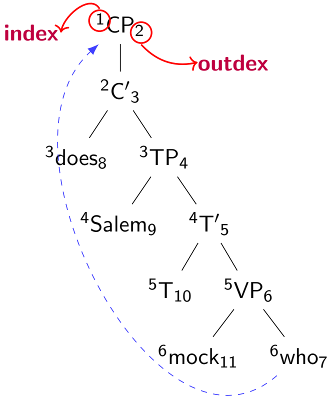
Incremental Top-Down Parsing

Technical details!

- String-driven recursive descent parser (Stabler 2013)

- • Who • does • Salem • T • mock

step 1 *CP* is conjectured
step 2 *CP* expands to *C'*
step 3 *C'* expands to *does* and *TP*
step 4 *TP* expands to *Salem* and *T'*
step 5 *T'* expands to *T* and *VP*
step 6 *VP* expands to *mock* and *who*
step 7 *who* is found
step 8 *does* is found
step 9 *Salem* is found
step 10 *T* is found
step 11 *mock* is found



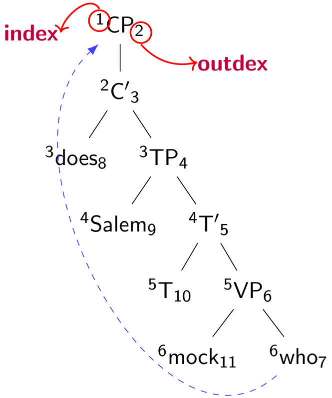
Incremental Top-Down Parsing

Technical details!

- String-driven recursive descent parser (Stabler 2013)

- • Who • does • Salem • T • mock

step 1	*CP* is conjectured
step 2	*CP* expands to *C'*
step 3	*C'* expands to *does* and *TP*
step 4	*TP* expands to *Salem* and *T'*
step 5	*T'* expands to *T* and *VP*
step 6	*VP* expands to *mock* and *who*
step 7	*who* is found
step 8	*does* is found
step 9	*Salem* is found
step 10	*T* is found
step 11	*mock* is found



Index and Outdex are our connection to memory!

Measuring Memory Usage

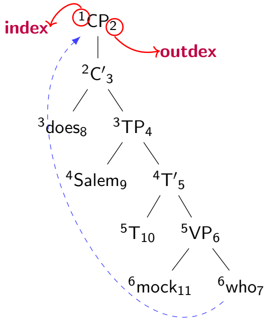
- **Memory usage**:
  (Kobele et al. 2012; Gibson, 1998)

Tenure How long a node is kept in memory

	Who	does	Salem	mock
Tenure	1	5	5	5

- Formalized into offline complexity metrics
  (Graf et a. 2017; De Santo 2020, 2021; a.o.)

MaxT $max(\{\text{tenure-of}(n) | n \text{ a node of the tree}\})$



Measuring Memory Usage

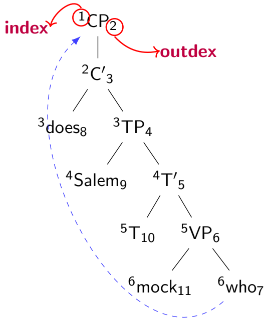
- **Memory usage**:
  (Kobele et al. 2012; Gibson, 1998)

Tenure How long a node is kept in memory

	Who	does	Salem	mock
Tenure	1	5	5	5

- Formalized into **offline complexity metrics**
  (Graf et a. 2017; De Santo 2020, 2021; a.o.)

MaxT $max(\{\text{tenure-of}(n)|n \text{ a node of the tree}\})$



Processing Asymmetries All the Way Down

A variety of offline processing insights!

Across Many Constructions

- Right > center embedding (Kobele et al. 2012)
- Crossing > nested dependencies (Kobele et al. 2012)
- SRC > ORC
  (Graf et al. 2017; De Santo 2020; Fiorini, Chang, De Santo 2023)
- Priming/Stacked RCs (De Santo 2020, 2022)
- Postverbal subjects
  (De Santo 2019, 2021; Del Valle & De Santo 2023)
- Persian attachment ambiguities (De Santo & Shafiei 2019)
- RC attachment preferences
  (De Santo & Lee 2022; Lee & De Santo 2023)

Across Languages

- English, German, Italian, French, Spanish
- Korean, Japanese, Mandarin Chinese
- Basque, Persian, ...

Processing Asymmetries All the Way Down

A variety of offline processing insights!

Across Many Constructions

- Right > center embedding (Kobele et al. 2012)
- Crossing > nested dependencies (Kobele et al. 2012)
- SRC > ORC
  (Graf et al. 2017; De Santo 2020; Fiorini, Chang, De Santo 2023)
- Priming/Stacked RCs (De Santo 2020, 2022)
- Postverbal subjects
  (De Santo 2019, 2021; Del Valle & De Santo 2023)
- Persian attachment ambiguities (De Santo & Shafiei 2019)
- RC attachment preferences
  (De Santo & Lee 2022; Lee & De Santo 2023)

Across Languages

- English, German, Italian, French, Spanish
- Korean, Japanese, Mandarin Chinese
- Basque, Persian, ...

A Case Study: English SRC vs ORC

(1) The horse that has chased the lions **SRC**

(2) The horse that the lions have chased **ORC**

SRC > ORC

- Well-attested cross-linguistically (Lau & Tanaka 2021)
- ... with some possible exceptions (Mandarin?)

Possible Accounts?

- Working-memory (Warren & Gibson 2008; Lewis & Vasishth, 2005; a.o.)
  $\Rightarrow$ BUT: Nakamura & Miyamoto 2(013) Cf. Graf et al (2017)
- Expectation-based accounts (Hale 2001; Demberg Keller, 2008; Chen & Hale 2021)
  $\Rightarrow$ BUT: Levy & Gibson (2013); Huang et al. (2024)

A Case Study: English SRC vs ORC

(1) The horse that has chased the lions **SRC**

(2) The horse that the lions have chased **ORC**

SRC > ORC

- Well-attested cross-linguistically (Lau & Tanaka 2021)
- ... with some possible exceptions (Mandarin?)

Possible Accounts?

- Working-memory
  (Warren & Gibson 2008; Lewis & Vasishth, 2005; a.o.)
  ⇒ BUT: Nakamura & Miyamoto 2(013) Cf. Graf et al (2017)
- Expectation-based accounts
  (Hale 2001; Demberg Keller, 2008; Chen & Hale 2021)
  ⇒ BUT: Levy & Gibson (2013); Huang et al. (2024)

A Case Study: English SRC vs ORC

(1) The horse that has chased the lions **SRC**

(2) The horse that the lions have chased **ORC**

SRC > ORC

- Well-attested cross-linguistically (Lau & Tanaka 2021)
- ... with some possible exceptions (Mandarin?)

Possible Accounts?

- Working-memory
  (Warren & Gibson 2008; Lewis & Vasishth, 2005; a.o.)
  ⇒ BUT: Nakamura & Miyamoto 2(013) Cf. Graf et al (2017)
- Expectation-based accounts
  (Hale 2001; Demberg Keller, 2008; Chen & Hale 2021)
  ⇒ BUT: Levy & Gibson (2013); Huang et al. (2024)

Modeling Assumptions

Data

- SAP Benchmark (Huang et al. 2024)
  - self-paced reading
  - 2000 participants
  - SRC/ORC RTs
  - 24 RC sets

Reminder: Model Details

- Parsing strategy
  ⇒ Top-down parser
- Linking Hypothesis
  ⇒ Processing Cost :: (word-by-word) Tenure

Degrees of freedom: Syntactic analyses

- RC constructions → (Kayne 1994)

Modeling Assumptions

Data

- SAP Benchmark (Huang et al. 2024)
  - self-paced reading
  - 2000 participants
  - SRC/ORC RTs
  - 24 RC sets

Reminder: Model Details

- Parsing strategy
  $\Rightarrow$ Top-down parser

- Linking Hypothesis
  $\Rightarrow$ Processing Cost :: (word-by-word) Tenure

Degrees of freedom: Syntactic analyses

- RC constructions $\rightarrow$ (Kayne 1994)

Modeling Assumptions

Data

- SAP Benchmark (Huang et al. 2024)
  - self-paced reading
  - 2000 participants
  - SRC/ORC RTs
  - 24 RC sets

Reminder: Model Details

- Parsing strategy
  $\Rightarrow$ Top-down parser
- Linking Hypothesis
  $\Rightarrow$ Processing Cost :: (word-by-word) Tenure

Degrees of freedom: Syntactic analyses

- RC constructions $\rightarrow$ (Kayne 1994)

Results: Model Comparison

Baseline Model (Huang et al. 2024)

$$RT \sim WordPosition(i) + logfreq(i) * length(i) + logfreq(i-1) * length(i-1) + logfreq(i-1) * length(i-2) + (1|participant) + (1|item)$$

	df	AIC	BIC
Baseline	14	977122.5	977250.8

Results: Model Comparison

Baseline Model (Huang et al. 2024)

$$RT \sim WordPosition(i) + logfreq(i) * length(i)$$
$$+ logfreq(i-1) * length(i-1) + logfreq(i-1) * length(i-2)$$
$$+ (1|participant) + (1|item)$$

	df	**AIC**	**BIC**
Baseline	14	977122.5	977250.8
+ LSTM Surprisal	19	976309.1	976483.1
+ GPT-2 Small Surprisal	19	976301.9	976475.9
+ Tenure	19	974413.7	974587.7

Results: Model Comparison

Baseline Model (Huang et al. 2024)

$$RT \sim WordPosition(i) + logfreq(i) * length(i) + logfreq(i-1) * length(i-1) + logfreq(i-1) * length(i-2) + (1|participant) + (1|item)$$

	df	AIC	BIC
Baseline	14	977122.5	977250.8
+ LSTM Surprisal	19	976309.1	976483.1
+ GPT-2 Small Surprisal	19	976301.9	976475.9
+ Tenure	19	974413.7	974587.7
+ LSTM Surprisal + Tenure	23	974174.8	974385.5
+ GPT Surprisal + Tenure	24	974106.3	974326.2

Results: Model Comparison

Baseline Model (Huang et al. 2024)

$$RT \sim WordPosition(i) + logfreq(i) * length(i) + logfreq(i-1) * length(i-1) + logfreq(i-1) * length(i-2) + (1|participant) + (1|item)$$

	df	AIC	BIC
Baseline	14	977122.5	977250.8
+ LSTM Surprisal	19	976309.1	976483.1
+ GPT-2 Small Surprisal	19	976301.9	976475.9
+ **Tenure**	**19**	**974413.7**	**974587.7**
+ LSTM Surprisal + Tenure	23	974174.8	974385.5
+ **GPT Surprisal + Tenure**	**24**	**974106.3**	**974326.2**

Results: Best Fitting Model

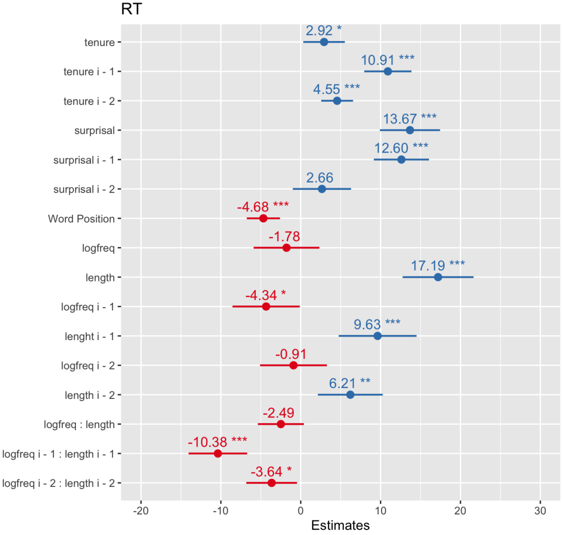


Figure: Estimates of coefficients for GTP Surprisal + Tenure.

Conclusion

TL;DR

MG-based Tenure is a good predictor of RTs.

- Support for MGs + Tenure beyond offline measures!
- Bridge generative syntax/sentence processing!
- Next: cross-linguistic online data, Tenure and empty heads...

The tip of the iceberg!

- Structure- vs. expectation-based predictors!
  (Demberg & Keller 2008; Brennan et al., 2016; Stanojevic et al., 2023; Ozaki et al. 2024)
- Deeper exploration of computational linking theories
  (Futrell et al., 2020; Chen and Hale, 2021; Oh et al., 2022; Arehalli et al., 2022; Kajikawa et al. 2024)
- Cross-formalism comparisons
- And much more!

Conclusion

TL;DR

MG-based Tenure is a good predictor of RTs.

- Support for MGs + Tenure beyond offline measures!
- Bridge generative syntax/sentence processing!
- Next: cross-linguistic online data, Tenure and empty heads...

The tip of the iceberg!

- Structure- vs. expectation-based predictors!
  (Demberg & Keller 2008; Brennan et al., 2016; Stanojevic et al., 2023; Ozaki et al. 2024)
- Deeper exploration of computational linking theories
  (Futrell et al., 2020; Chen and Hale, 2021; Oh et al., 2022; Arehalli et al., 2022; Kajikawa et al. 2024)
- Cross-formalism comparisons
- And much more!

Thank you!

Appendix

Minimalist Grammars (MGs)

We need an explicit model of syntactic structures...

Ed Stabler

- Minimalist grammars (**MGs**): a formalization of Chomskyan syntax (Chomsky 1995; Stabler 1997)

Technical details!

- Weakly equivalent to MCFGs
- Essentially: CFGs with a more complicated mapping from trees to strings
- REG tree language!

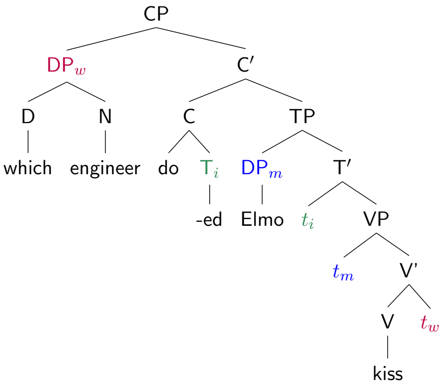
Why MGs?

1. Vast analytical coverage
   - MGs handle virtually all analyses in the generative literature
2. Centrality of derivation trees
   - MGs can be viewed as CFGs with a more complicated mapping from trees to strings
3. Simple parsing algorithms
   - Variant of a recursive descent parser for CFGs
     $\Rightarrow$ cf. TAG (Rambow & Joshi, 1995; Demberg, 2008)

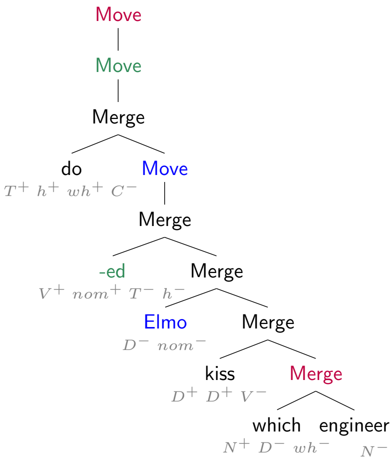
Some Important Properties of MGs

- MGs are weakly equivalent to MCFGs and thus mildly context-sensitive. (Harkema 2001, Michaelis 2001)
- But we can decompose them into two finite-state components: (Michaelis et al. 2001, Kobele et al. 2007, Monnich 2006)
  - a regular language of well-formed derivation trees
  - an MSO-definable mapping from derivations to phrase structure trees
- **Remember:** Every regular tree language can be re-encoded as a CFG (with more fine-grained non-terminal labels). (Thatcher 1967)

Fully Specified Derivation Trees



Phrase Structure Tree



Derivation Tree

Technical Fertility of MGs

MGs can accommodate the full syntactic toolbox:

- sidewards movement (Stabler, 2006; Graf 2013)
- affix hopping (Graf 2012; Graf2013)
- clustering movement (Gartner & Michaelis 2010)
- tucking in (Graf 2013)
- ATB movement (Kobele 2008)
- copy movement (Kobele 2006)
- extraposition (Hunter &Frank 2014)
- Late Merge (Kobele 2010; Graf 2014)
- Agree (Kobele 2011; Graf 2011)
- adjunction (Fowlie 2013; Hunter 2015)
- TAG-style adjunction (Graf 2012)

Why These Metrics?

- These complexity metrics are all related to **storage cost** (cf. Gibson, 1998)
- We could implement alternative ones (cf. Ferrara-Boston, 2012)
  - number of bounding nodes / phases
  - surprisal
  - feature intervention
  - status of discourse referents
  - integration, retrieval, ...
- We want to keep the model **simple** (but not **trivial**):
  - Tenure and Size only refer to the geometry of the derivation
  - they are sensitive the specifics of tree-traversal (cf. node-count; Hale, 2001)

Why These Metrics?

- These complexity metrics are all related to **storage cost** (cf. Gibson, 1998)
- We could implement alternative ones (cf. Ferrara-Boston, 2012)
  - number of bounding nodes / phases
  - surprisal
  - feature intervention
  - status of discourse referents
  - integration, retrieval, ...
- We want to keep the model **simple** (but not **trivial**):
  - Tenure and Size only refer to the geometry of the derivation
  - they are sensitive the specifics of tree-traversal (cf. node-count; Hale, 2001)