Productivity, Reuse, and Competition between Generalizations

Timothy J. O'Donnell
MIT

Two Problems

1. Problem of Competition
2. Problem of Productivity

The Problem of Competition

When multiple ways of expressing a meaning exist, how do we decide between them?

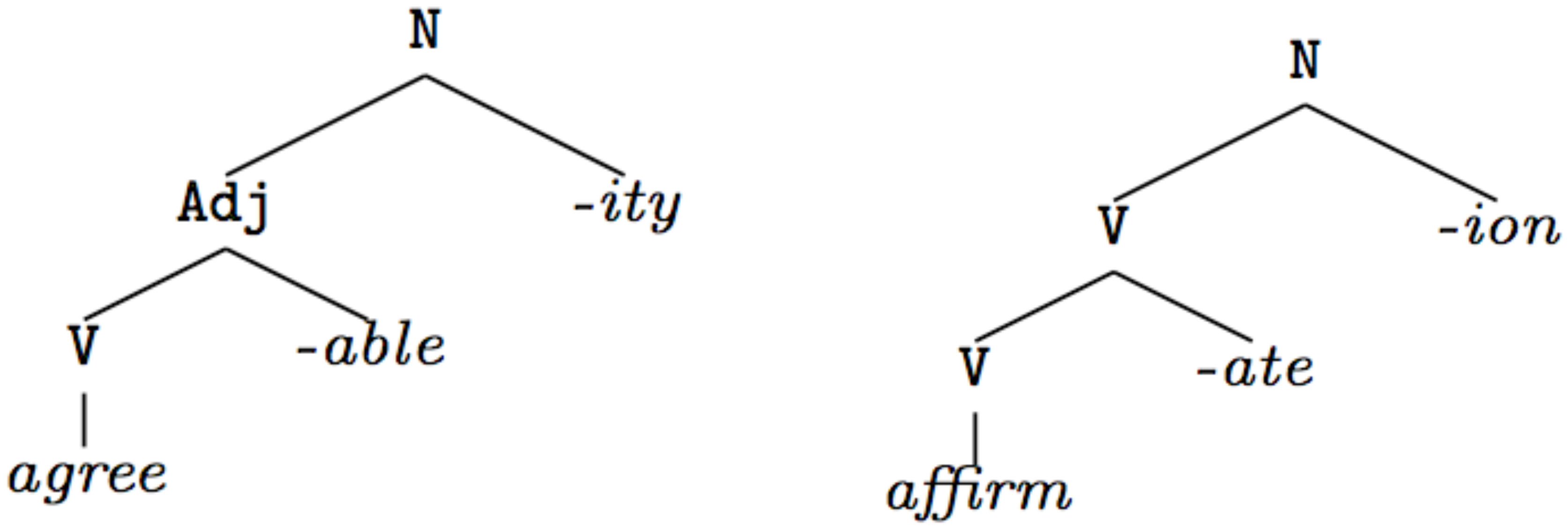
Competition

(e.g., Aronoff, 1976; Plag, 2003; Rainer, 1988; van Marle, 1986)

- Examples
  - Computed v. Stored
    - *goed* v. *went*
  - Computed v. Computed
    - *splinged* v. *splang* (Albright & Hayes, 2003)
  - Multi-way competition

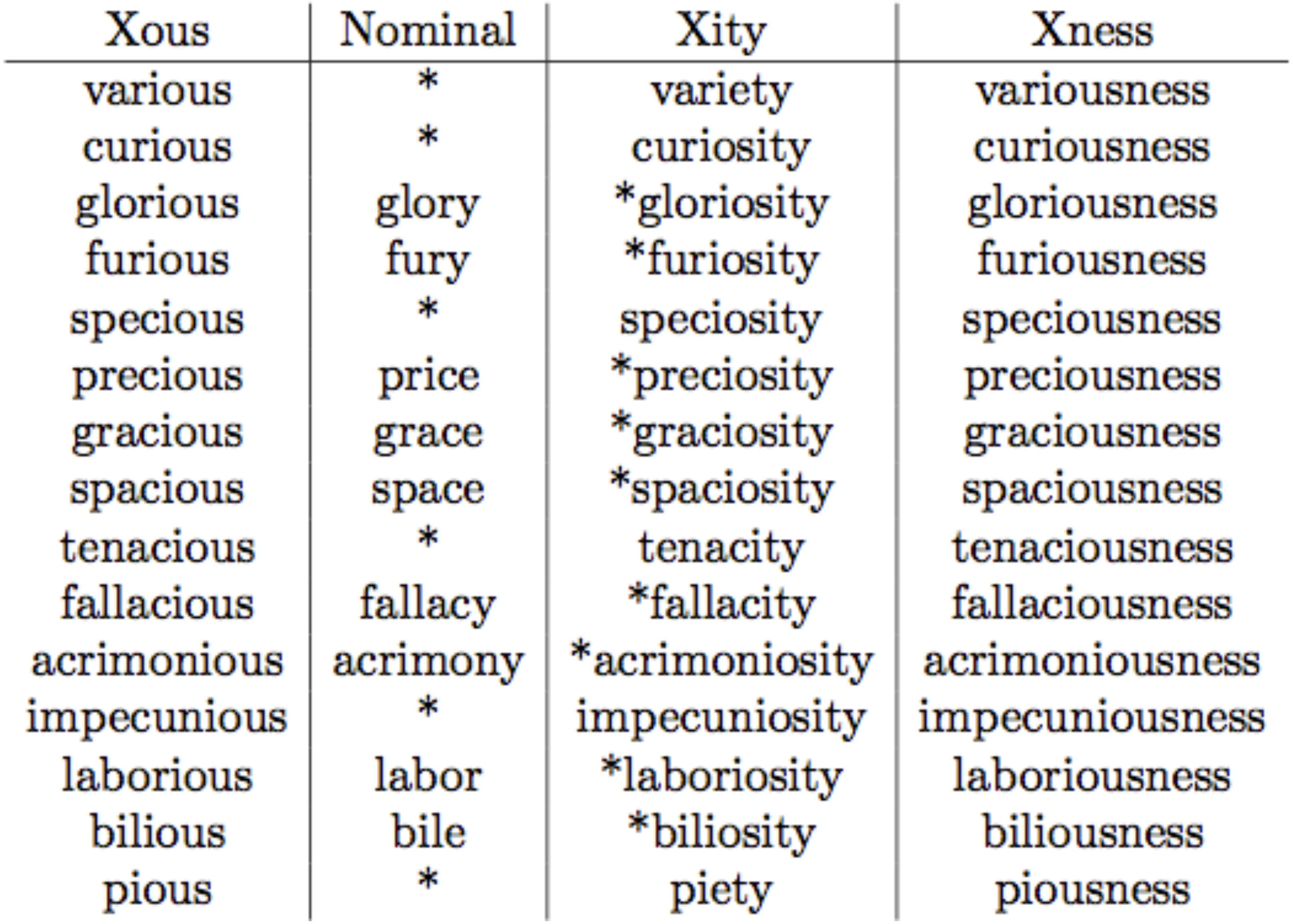
Multi-way Competition

- Hierarchical and recursive structures often give rise to multi-way competition between different combinations of stored and computed subexpression.



Multi-way Competition

(Aronoff, 1976)



Xous	Nominal	Xity	Xness
various	*	variety	variousness
curious	*	curiosity	curiousness
glorious	glory	*gloriosity	gloriousness
furious	fury	*furiosity	furiousness
specious	*	speciosity	speciousness
precious	price	*preciosity	preciousness
gracious	grace	*graciosity	graciousness
spacious	space	*spaciosity	spaciousness
tenacious	*	tenacity	tenaciousness
fallacious	fallacy	*fallacity	fallaciousness
acrimonious	acrimony	*acrimoniosity	acrimoniousness
impecunious	*	impecuniosity	impecuniousness
laborious	labor	*laboriosity	laboriousness
bilious	bile	*biliosity	biliousness
pious	*	piety	piousness

Competition Resolution

- Competition is resolved in general following the *elsewhere condition* (*subset principle*, *Pāṇini's principle*, *blocking*, *pre-emption*, etc.)
    - "More specific" way of expressing meaning is preferred to "more general" way.
- Variability in strength of preferences
    - *goed* v. *went*
    - *curiosity* v. *curiousness, depulsiveness v. depulsivity* (Aronoff & Schvaneveldt, 1978)
    - *tolerance* v. *toleration* (i.e., *doublets*, e.g., Kiparsky, 1982a)
- More frequent items are more strongly preferred (e.g., Marcus et al. 1992)

The Problem of Productivity

Why can some potential generalizations actually generalize productively, while others remain "inert" in existing expressions?

Productivity

	Suffix
Productive (with Adjectives)	-ness
Context-Dependent	-ity
Unproductive	-th

Productivity

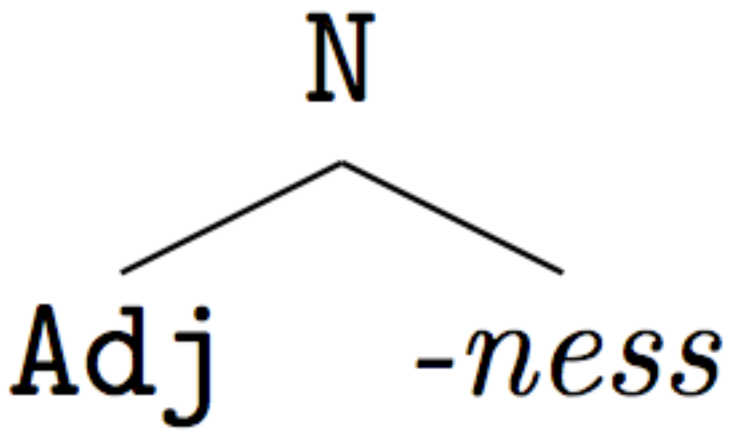
	Suffix
Productive (with Adjectives)	-ness

Existing: *circuitousness, grandness, orderliness, pretentiousness, cheapness, ...*

Novel: *pine-scented* *pine-scentedness*

Productivity

	Suffix
Productive (with Adjectives)	-ness



Productivity

	Suffix
Productive (with Adjectives)	-ness
Context-Dependent	-ity

Existing: *verticality,tractability,severity, seniority, inanity, electricity*, ...

Novel: **pine-scentedity*

Productivity

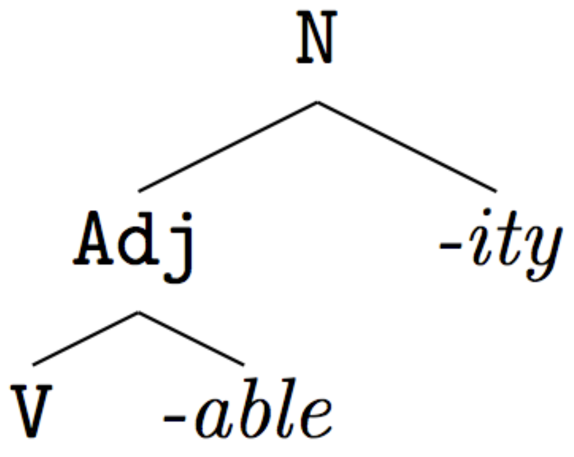
	Suffix
Productive (with Adjectives)	-ness
Context-Dependent	-ity

-ile, -al, -able, -ic, -(i)an

subsequentiable *subsequentiability*

Productivity

	Suffix
Productive (with Adjectives)	-ness
Context-Dependent	-ity



Productivity

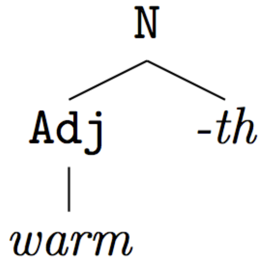
	Suffix
Productive (with Adjectives)	-ness
Context-Dependent	-ity
Unproductive	-th

Existing: *warmth*, *width*, *truth*, *depth*, ...

Novel: **coolth*

Productivity

	Suffix
Productive (with Adjectives)	-ness
Context-Dependent	-ity
Unproductive	-th



Productivity and Reuse

1. How can differences in productivity be represented?

2. How can differences be learned?

	Suffix
Most Productive	-ness
Less Productive	-ity
Least Productive	-th

Unifying the Problems

- **Fundamental problem**: How to produce/ comprehend linguistic expressions under uncertainty about how meaning is conventionally encoded by combinations of stored items and composed structures.
- Productivity and competition are often just special cases of this general problem.

Approach

- Build a model of computation and storage under uncertainty based on an inference which optimizes a **tradeoff** between **productivity** (computation) and **reuse** (storage).
- This implicitly explains many specific cases of productivity and competition.

Case Studies

1. What distributional factors signal productivity?
   - Explaining Baayen’s *hapax*-based measures.
2. How is competition resolved?
   - Derives *elsewhere condition*.
3. Multi-way competition.
   - Explains *productivity and ordering generalization*.
   - Handles exceptional cases of *paradoxical suffix combinations*.

Talk Outline

1. Introduction to productivity and reuse with Fragment Grammars (with Noah Goodman).
2. Case Studies on Productivity and Competition.

Talk Outline

1. Introduction to productivity and reuse with Fragment Grammars (with Noah Goodman).
2. Case Studies on Productivity and Competition.

The Framework: Three Ideas

1. Model how expressions are built by composing stored pieces.
2. Treat productivity (computation) and reuse (storage) as properties which must be determined on a case-by-case basis.
3. Infer correct patterns of storage and computation by balancing ability to predict input data against simplicity biases.

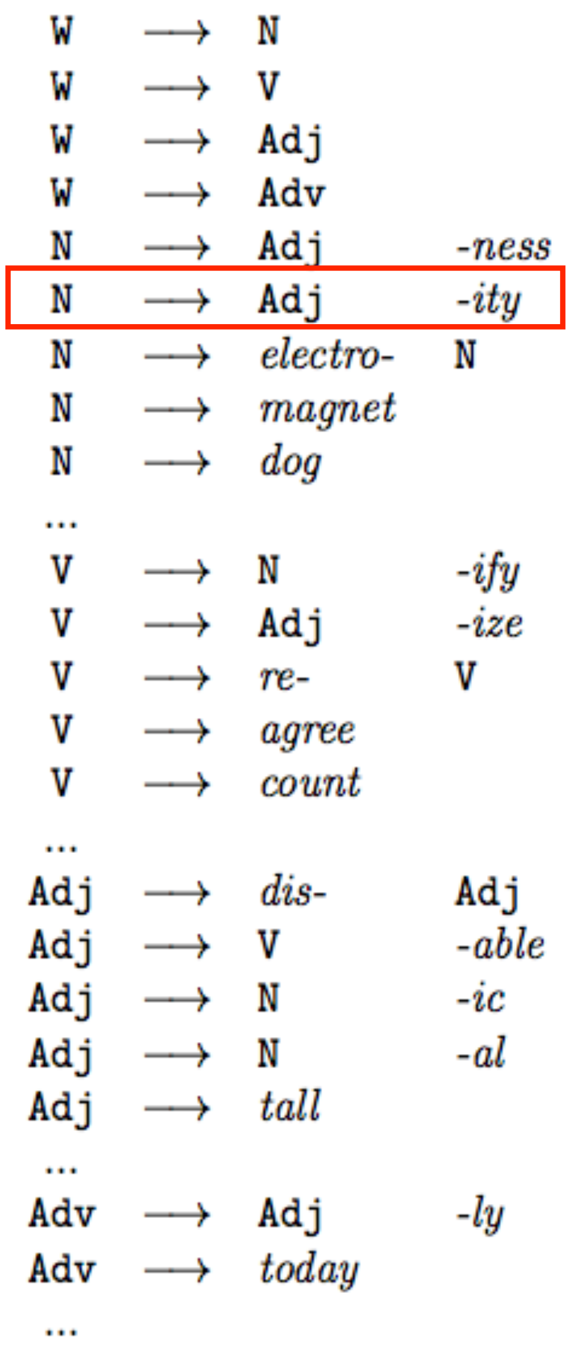
A Simple Formal Model: Fragment Grammars

1. Formalization of the hypothesis space.
   - Arbitrary contiguous (sub)trees.
2. Formalization of the inference problem.
   - Probabilistic conditioning to find good balance between computation and storage.

A Simple Formal Model: Fragment Grammars

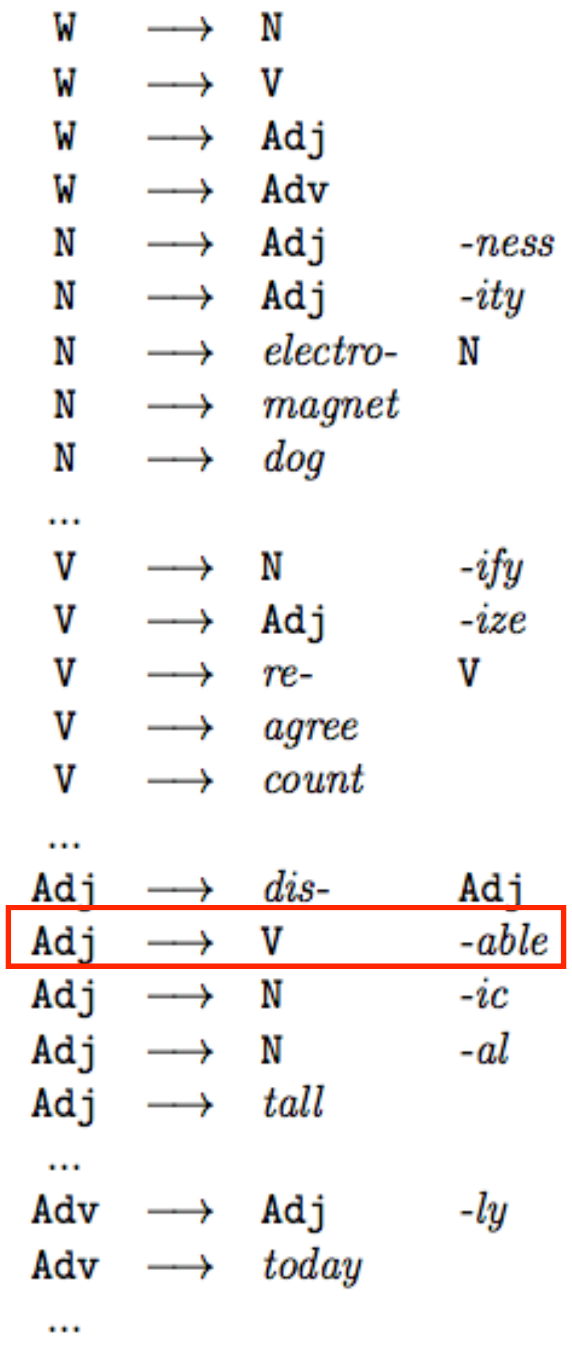
1. Formalization of the hypothesis space.
    - Arbitrary contiguous (sub)trees.
2. Formalization of the inference problem.
    - Probabilistic conditioning to find good balance between computation and storage.

Underlying Computational System

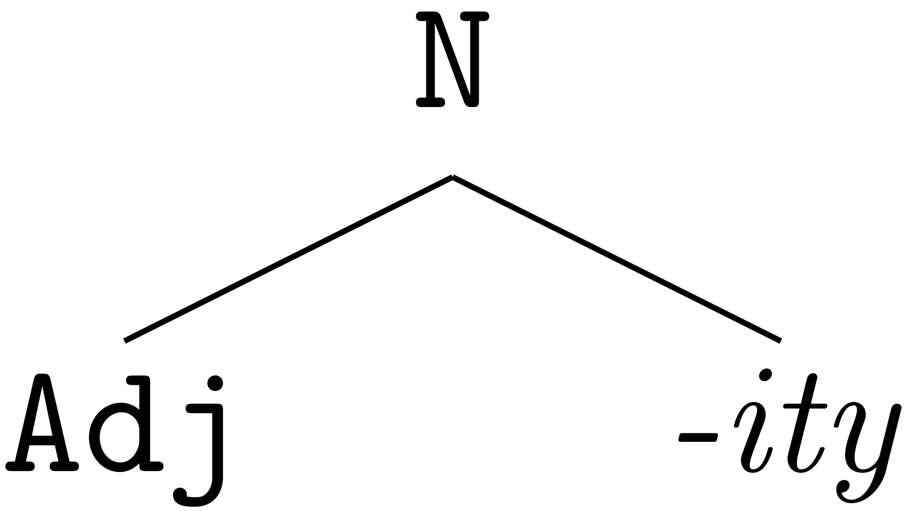


W	⟶	N	
W	⟶	V	
W	⟶	Adj	
W	⟶	Adv	
N	⟶	Adj	*-ness*
N	⟶	Adj	*-ity*
N	⟶	*electro-*	N
N	⟶	*magnet*	
N	⟶	*dog*	
...			
V	⟶	N	*-ify*
V	⟶	Adj	*-ize*
V	⟶	*re-*	V
V	⟶	*agree*	
V	⟶	*count*	
...			
Adj	⟶	*dis-*	Adj
Adj	⟶	V	*-able*
Adj	⟶	N	*-ic*
Adj	⟶	N	*-al*
Adj	⟶	*tall*	
...			
Adv	⟶	Adj	*-ly*
Adv	⟶	*today*	
...			

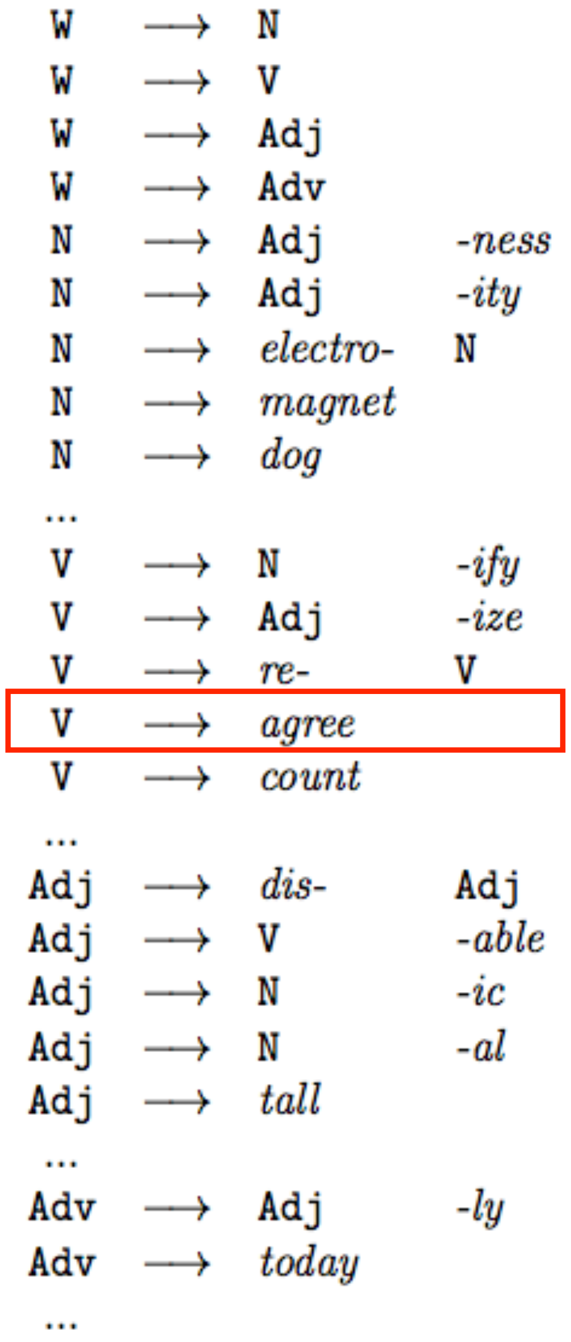
Underlying Computational System



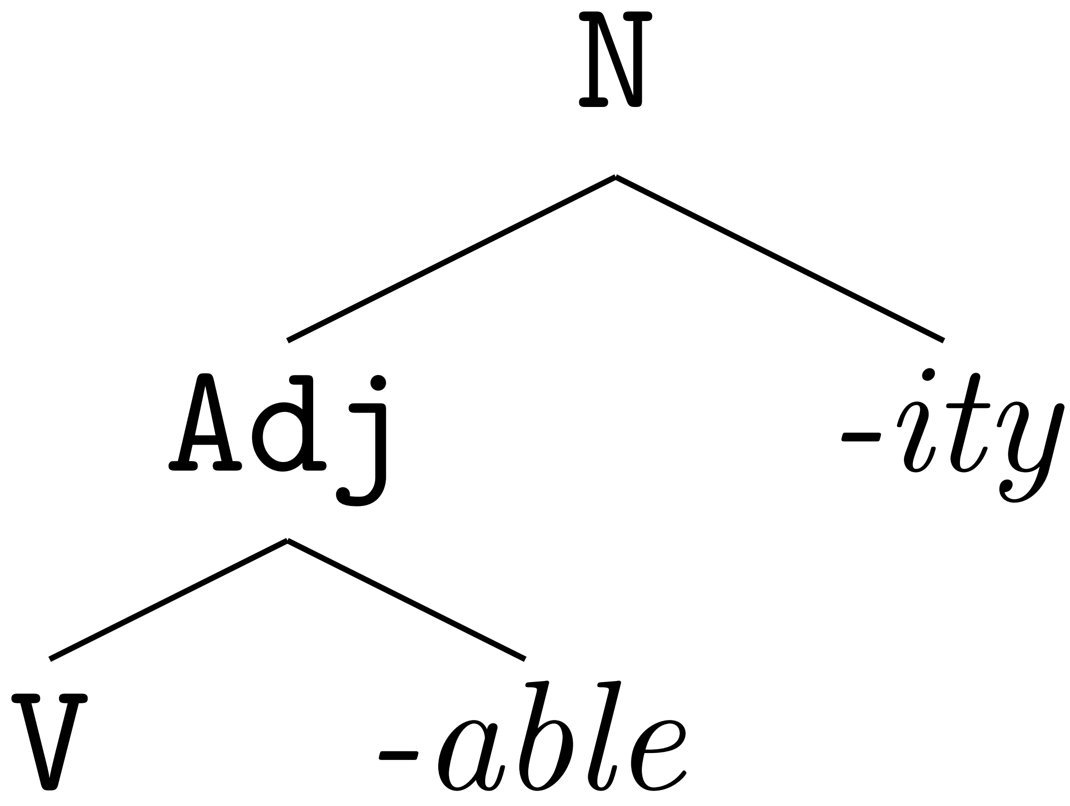
W	⟶	N	
W	⟶	V	
W	⟶	Adj	
W	⟶	Adv	
N	⟶	Adj	*-ness*
N	⟶	Adj	*-ity*
N	⟶	*electro-*	N
N	⟶	*magnet*	
N	⟶	*dog*	
...			
V	⟶	N	*-ify*
V	⟶	Adj	*-ize*
V	⟶	*re-*	V
V	⟶	*agree*	
V	⟶	*count*	
...			
Adj	⟶	*dis-*	Adj
Adj	⟶	V	*-able*
Adj	⟶	N	*-ic*
Adj	⟶	N	*-al*
Adj	⟶	*tall*	
...			
Adv	⟶	Adj	*-ly*
Adv	⟶	*today*	
...			



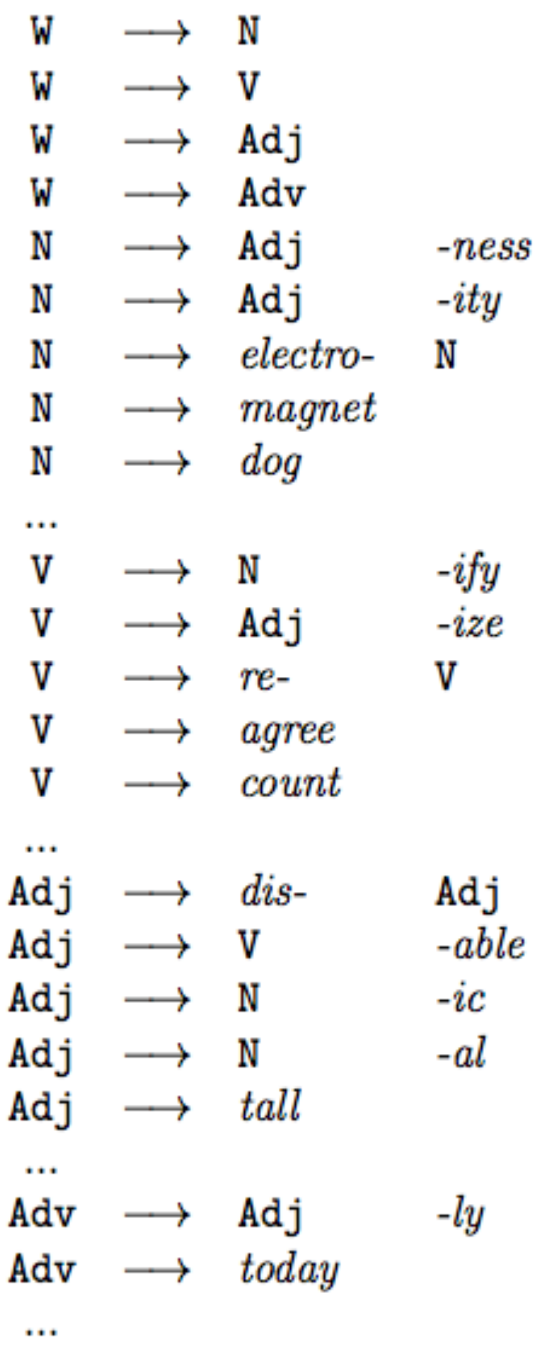
Underlying Computational System



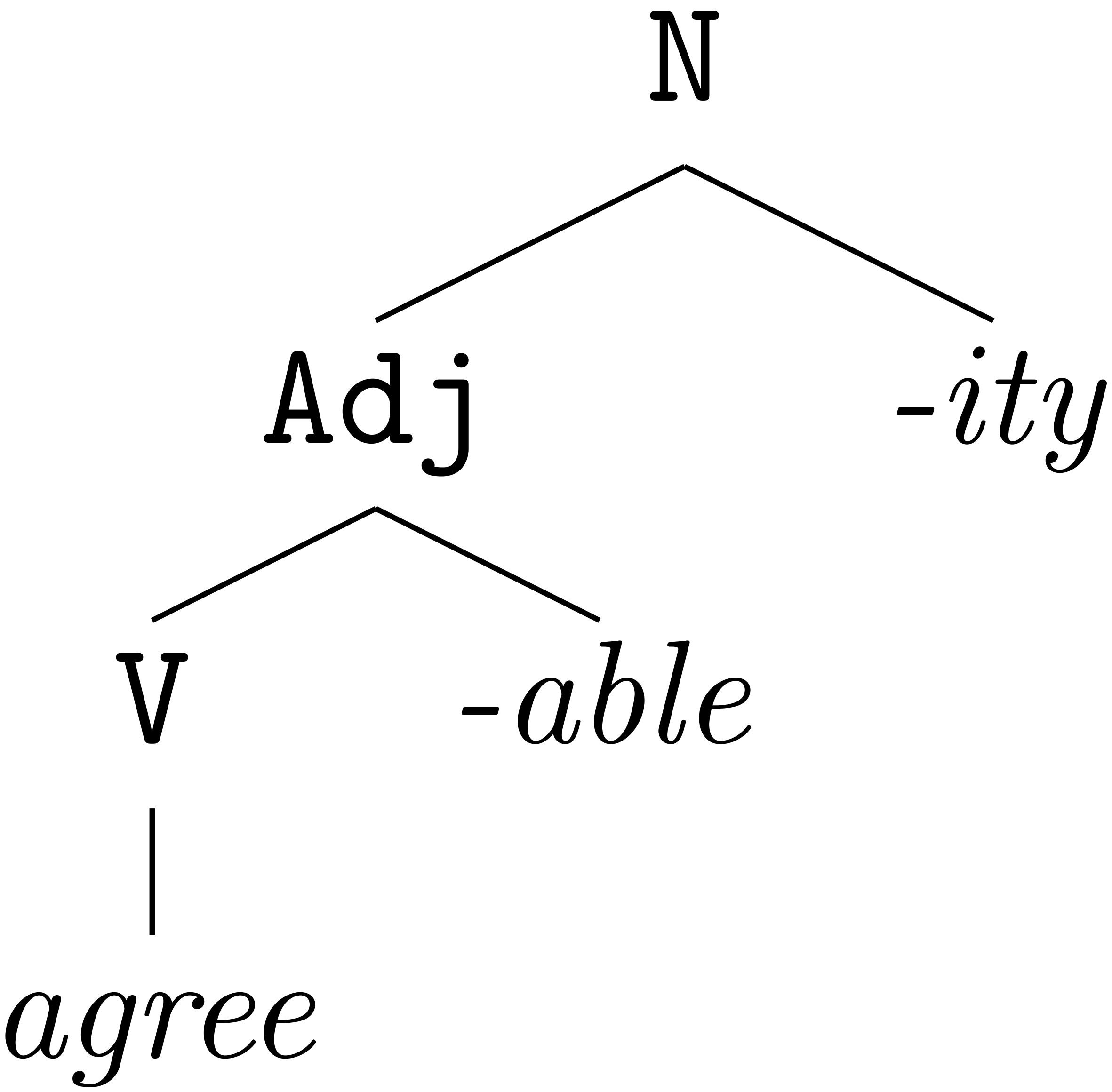
W	⟶	N	
W	⟶	V	
W	⟶	Adj	
W	⟶	Adv	
N	⟶	Adj	*-ness*
N	⟶	Adj	*-ity*
N	⟶	*electro-*	N
N	⟶	*magnet*	
N	⟶	*dog*	
...			
V	⟶	N	*-ify*
V	⟶	Adj	*-ize*
V	⟶	*re-*	V
V	⟶	*agree*	
V	⟶	*count*	
...			
Adj	⟶	*dis-*	Adj
Adj	⟶	V	*-able*
Adj	⟶	N	*-ic*
Adj	⟶	N	*-al*
Adj	⟶	*tall*	
...			
Adv	⟶	Adj	*-ly*
Adv	⟶	*today*	
...			



Underlying Computational System



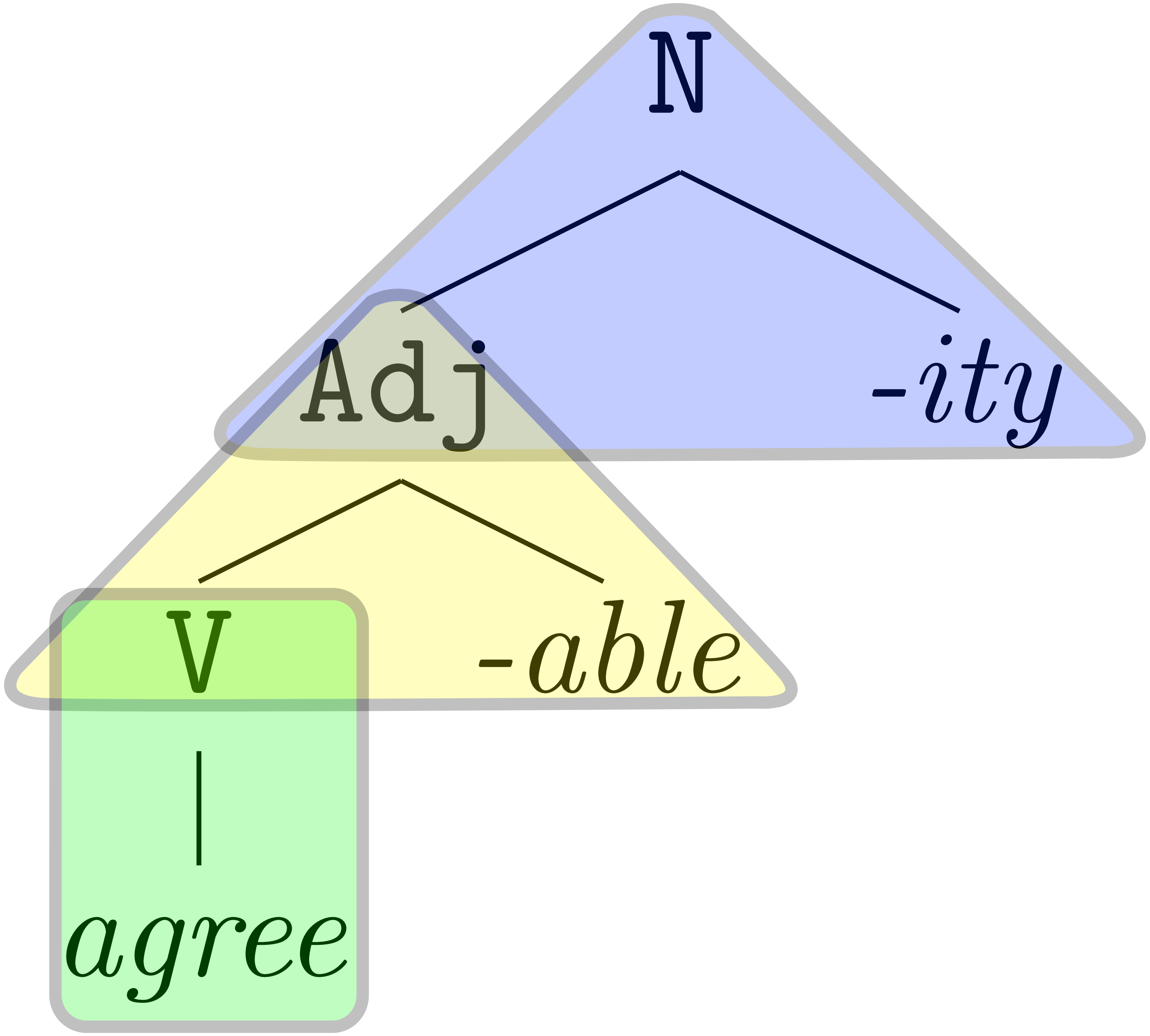
W	⟶	N	
W	⟶	V	
W	⟶	Adj	
W	⟶	Adv	
N	⟶	Adj	*-ness*
N	⟶	Adj	*-ity*
N	⟶	*electro-*	N
N	⟶	*magnet*	
N	⟶	*dog*	
...			
V	⟶	N	*-ify*
V	⟶	Adj	*-ize*
V	⟶	*re-*	V
V	⟶	*agree*	
V	⟶	*count*	
...			
Adj	⟶	*dis-*	Adj
Adj	⟶	V	*-able*
Adj	⟶	N	*-ic*
Adj	⟶	N	*-al*
Adj	⟶	*tall*	
...			
Adv	⟶	Adj	*-ly*
Adv	⟶	*today*	
...			



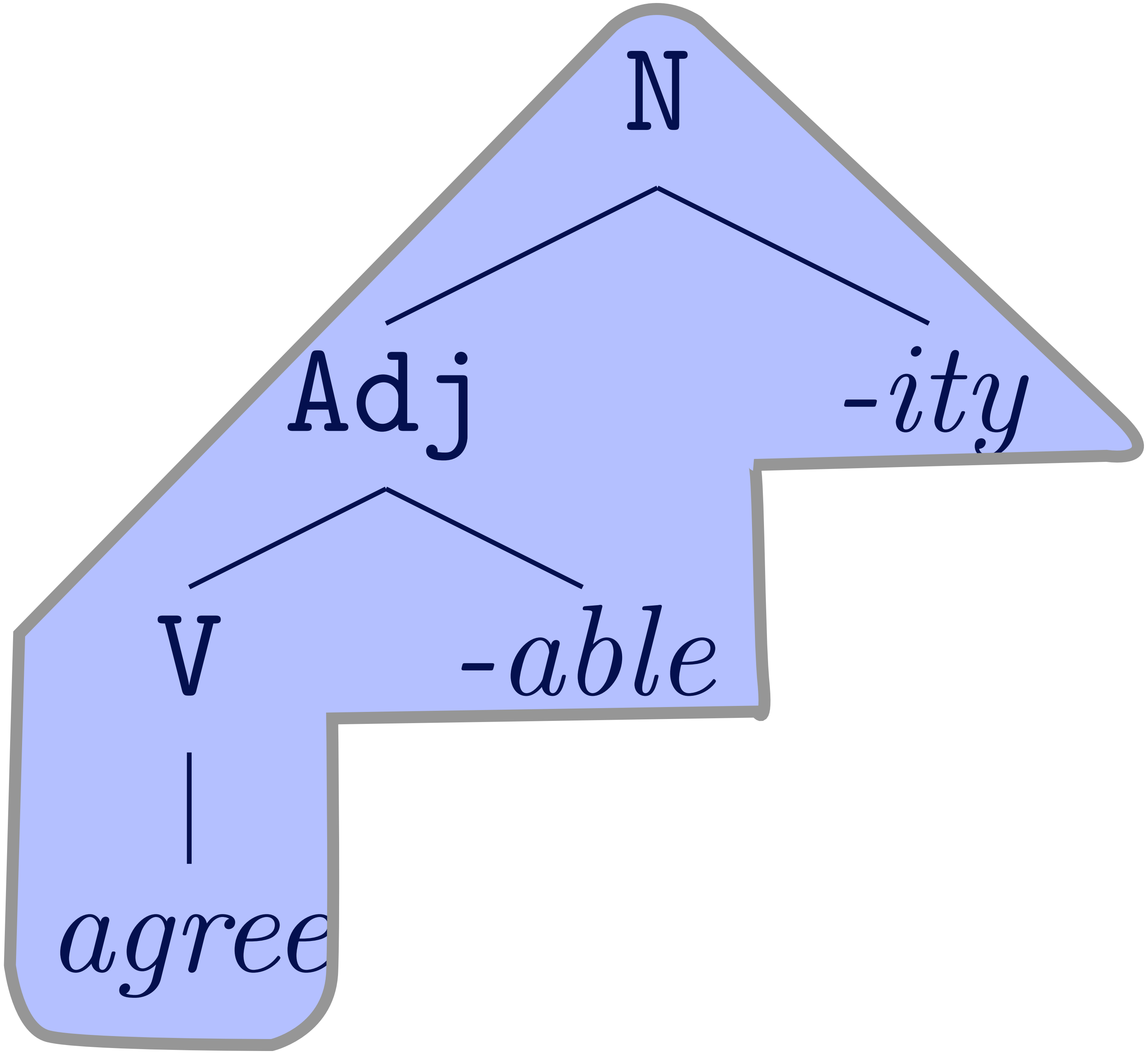
Hypothesis Space

Any contiguous subtree can be stored in memory and reused as if it were a single rule from the starting grammar.

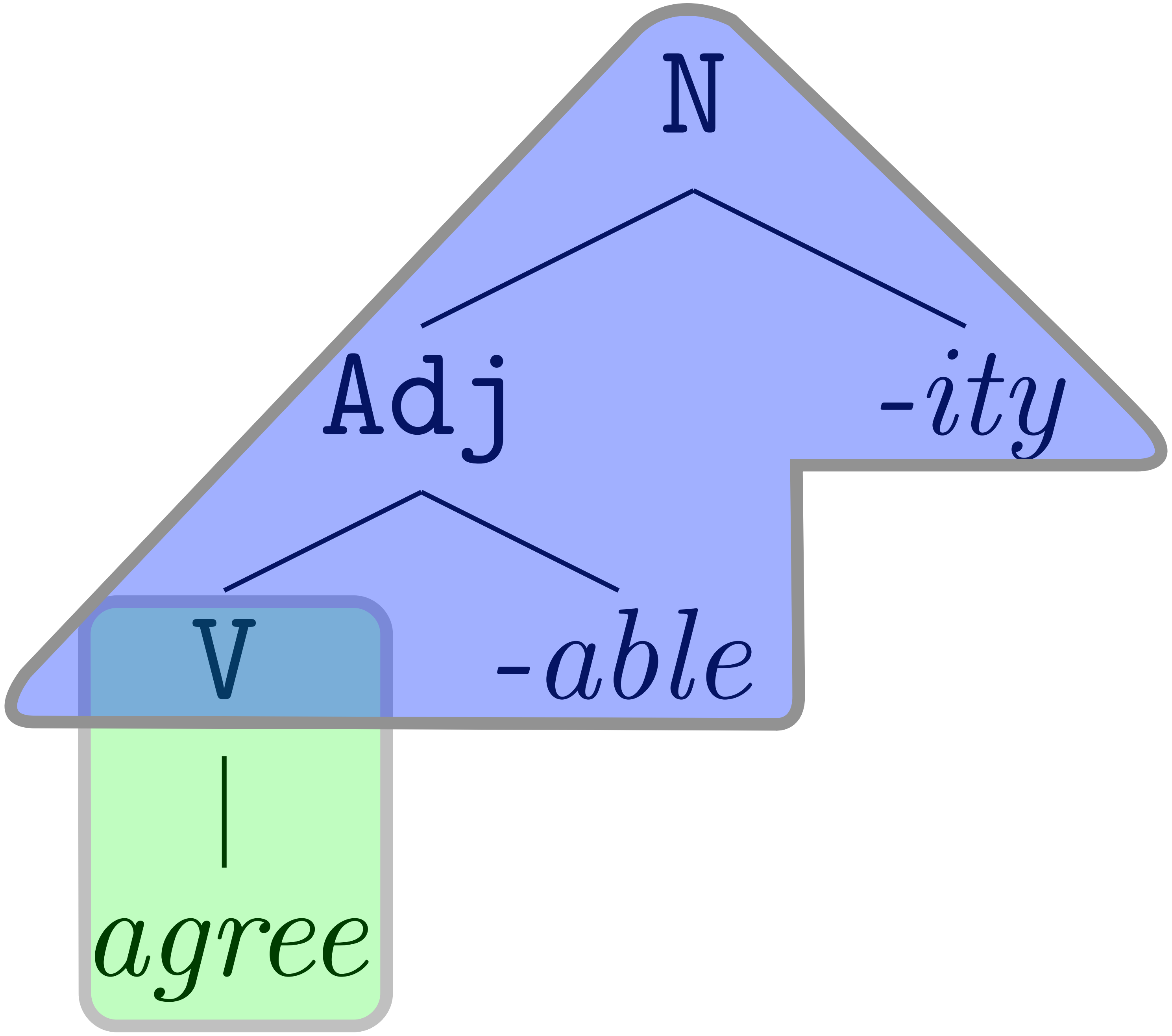
Hypothesis Space



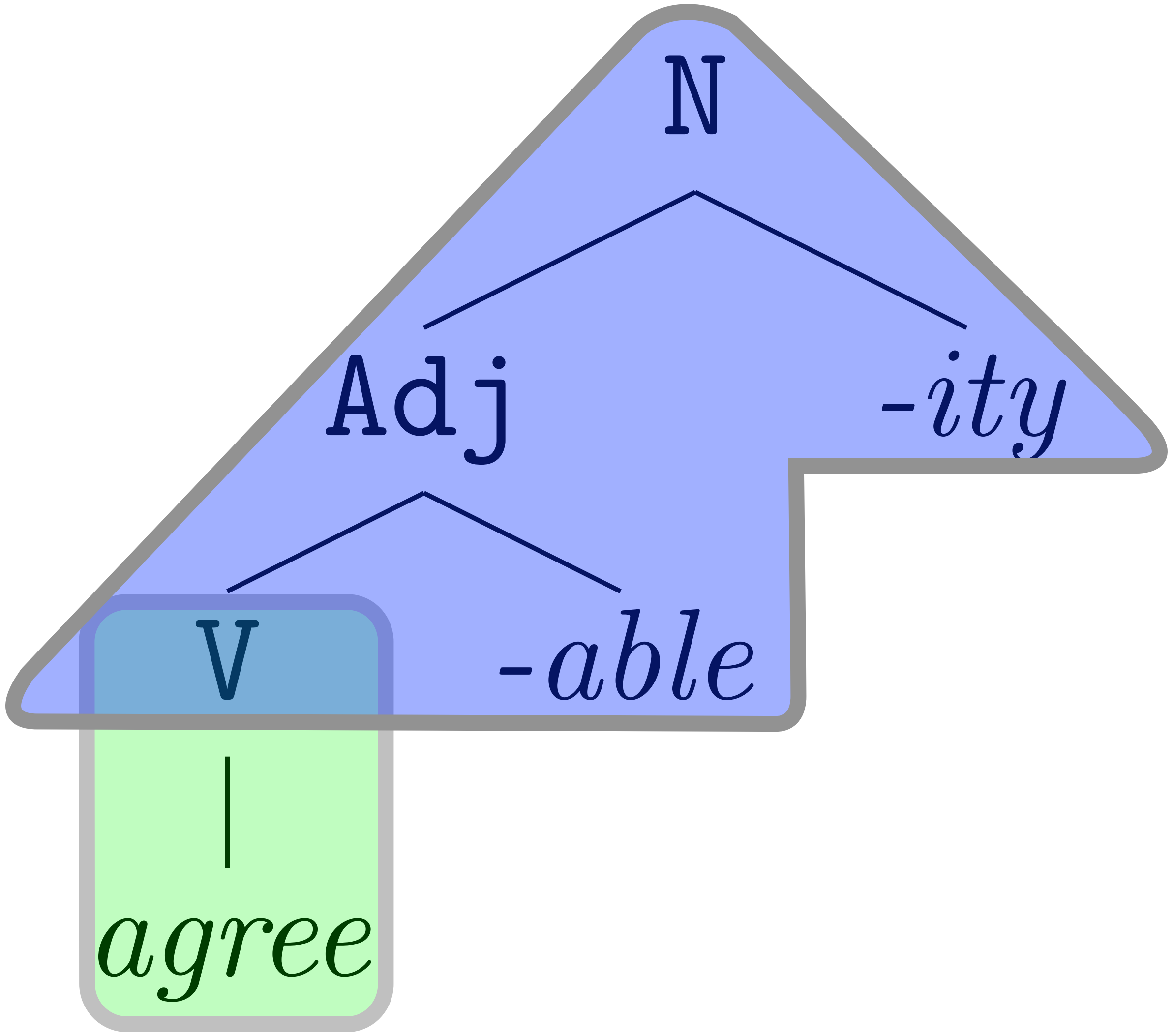
Hypothesis Space



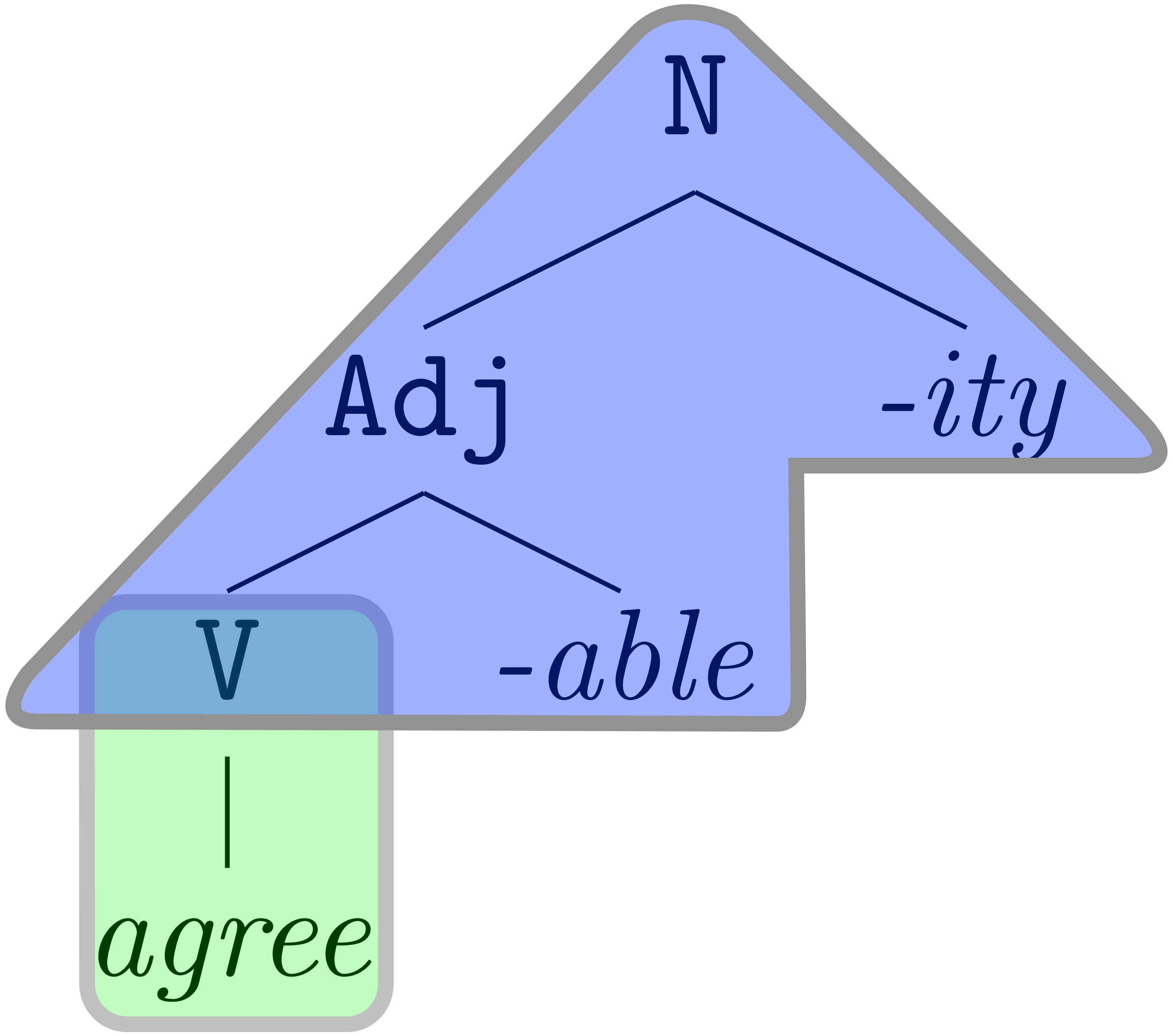
Hypothesis Space

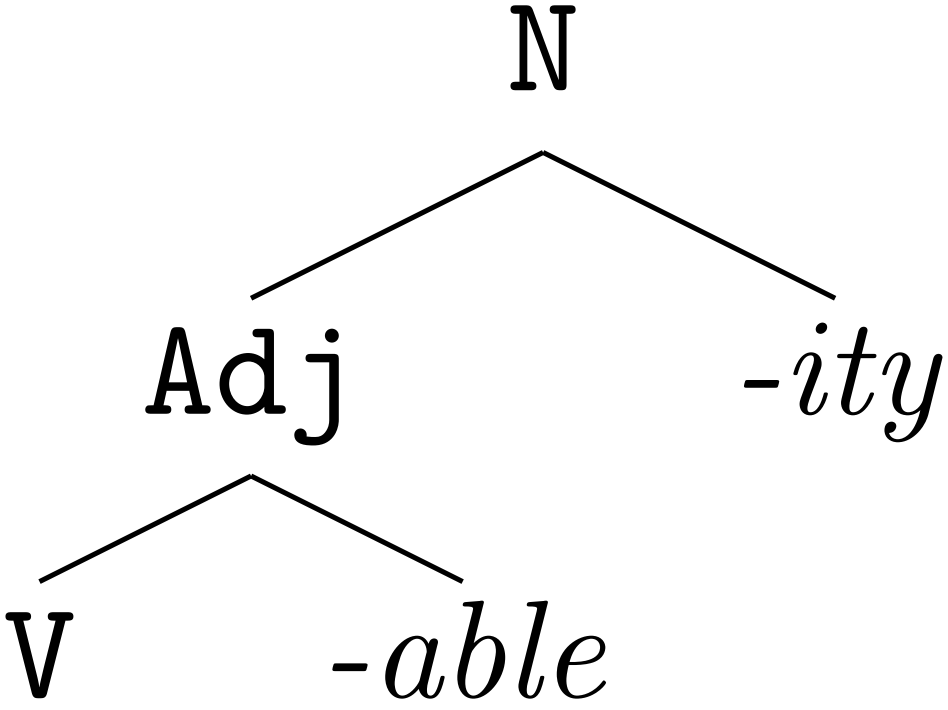


Computation with Stored items

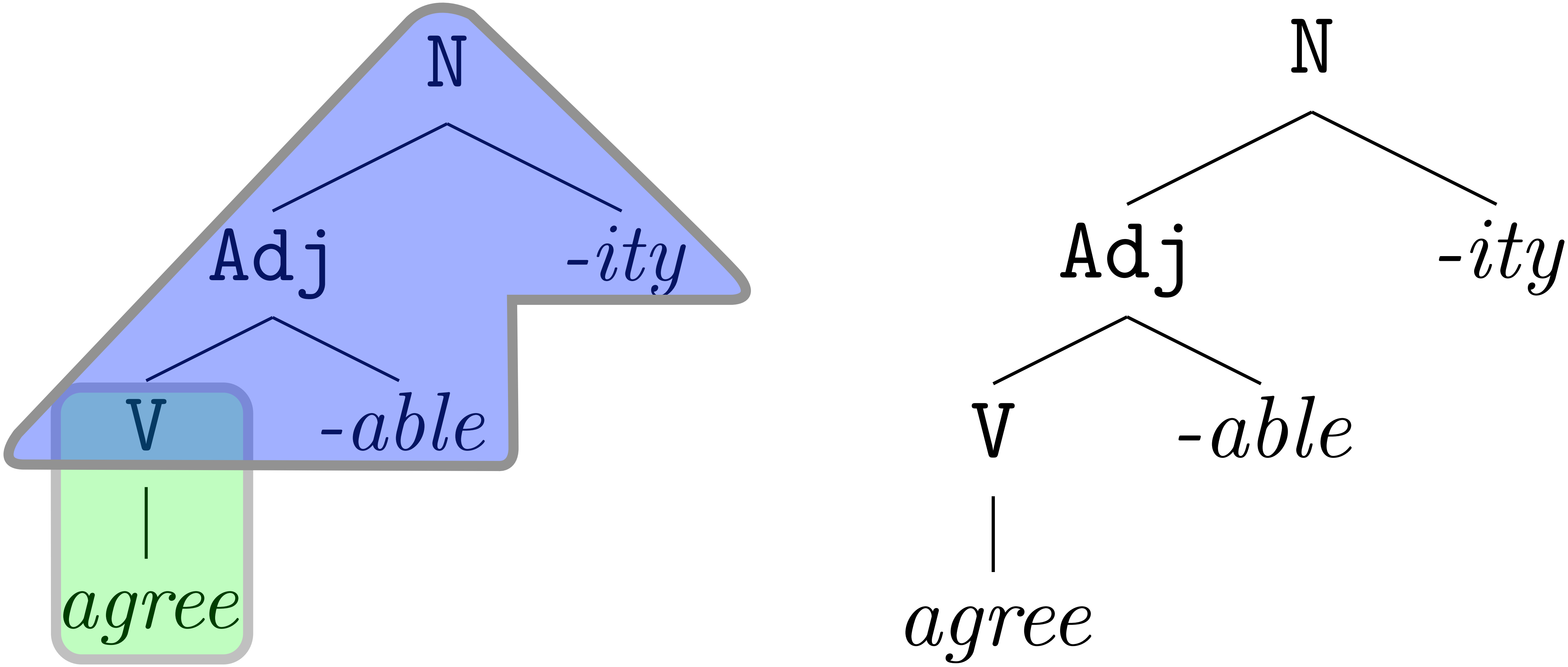


Computation with Stored items





Computation with Stored items



A Simple Formal Model: Fragment Grammars

1. Formalization of the hypothesis space.
    - Arbitrary contiguous (sub)trees.
2. Formalization of the inference problem.
    - Probabilistic conditioning to find good balance between computation and storage.

Inference Problem

Find and store the subcomputations which best **predict** the distribution of forms in the linguistic input taking into account **prior** expectations for simplicity.

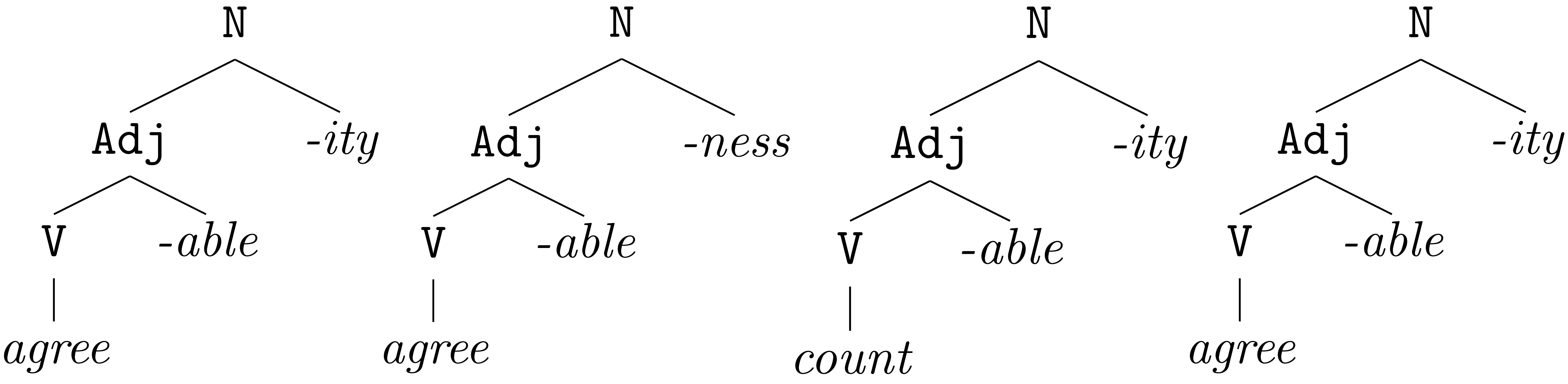
Prior Expectations

Two Opposing Simplicity Biases

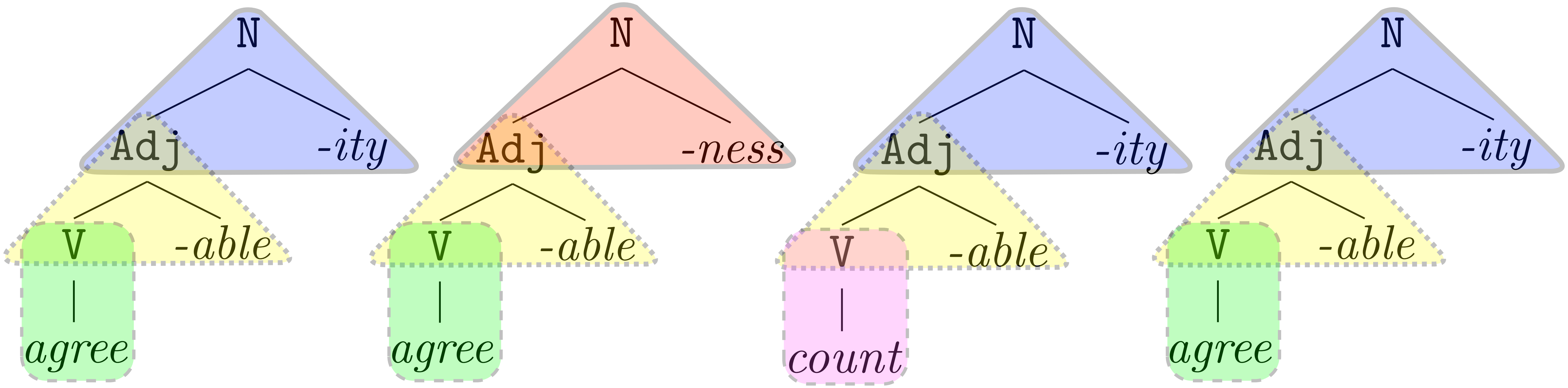
1. Fewer, more reusable stored items.
   - Chinese Restaurant process prior on lexica.

2. Small amounts of computation.
   - Geometric decrease in probability in number of random choices.

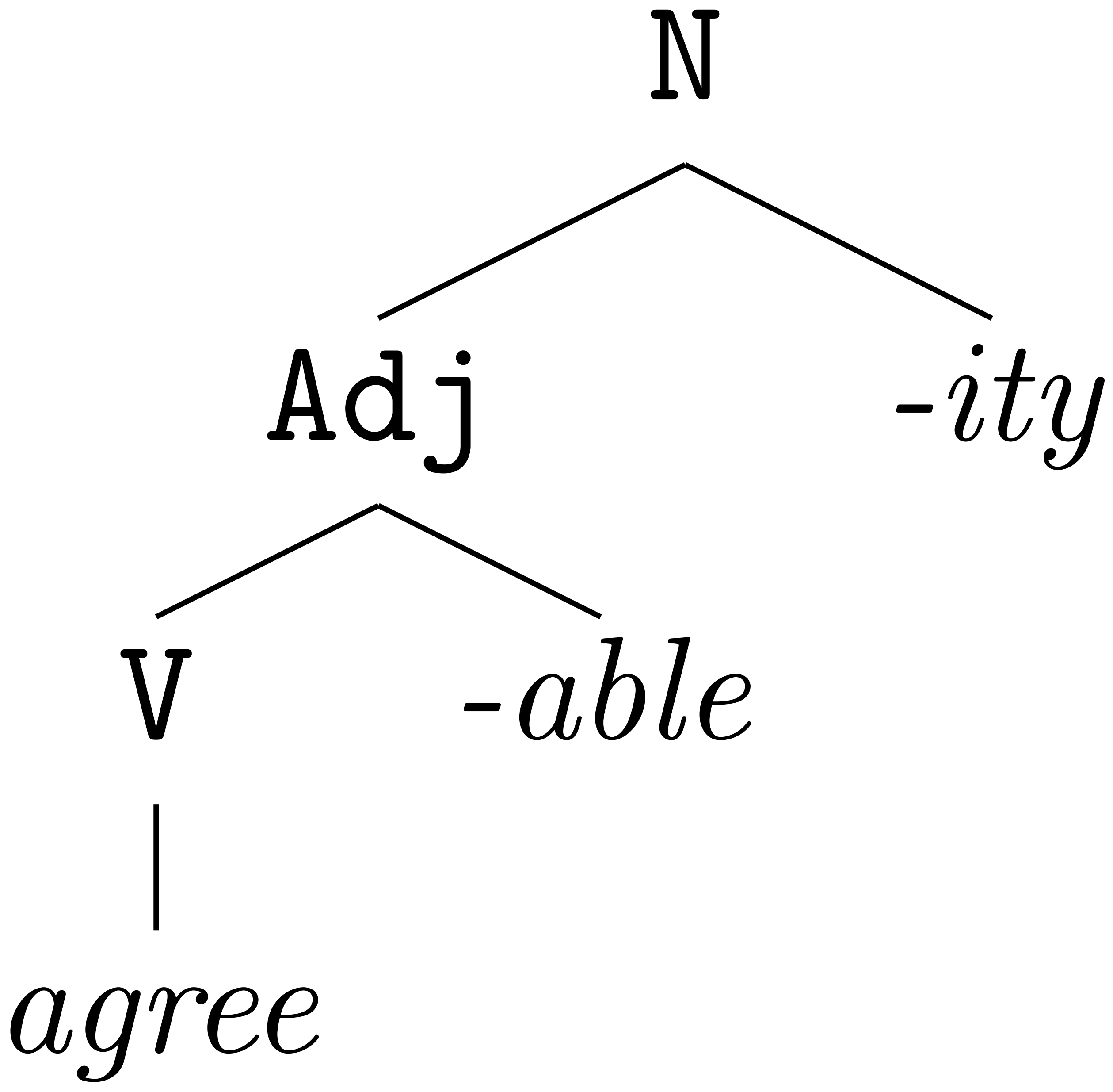
Example Input



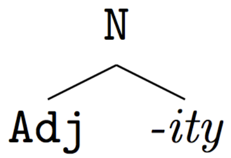
Storage of Minimal, General Structures

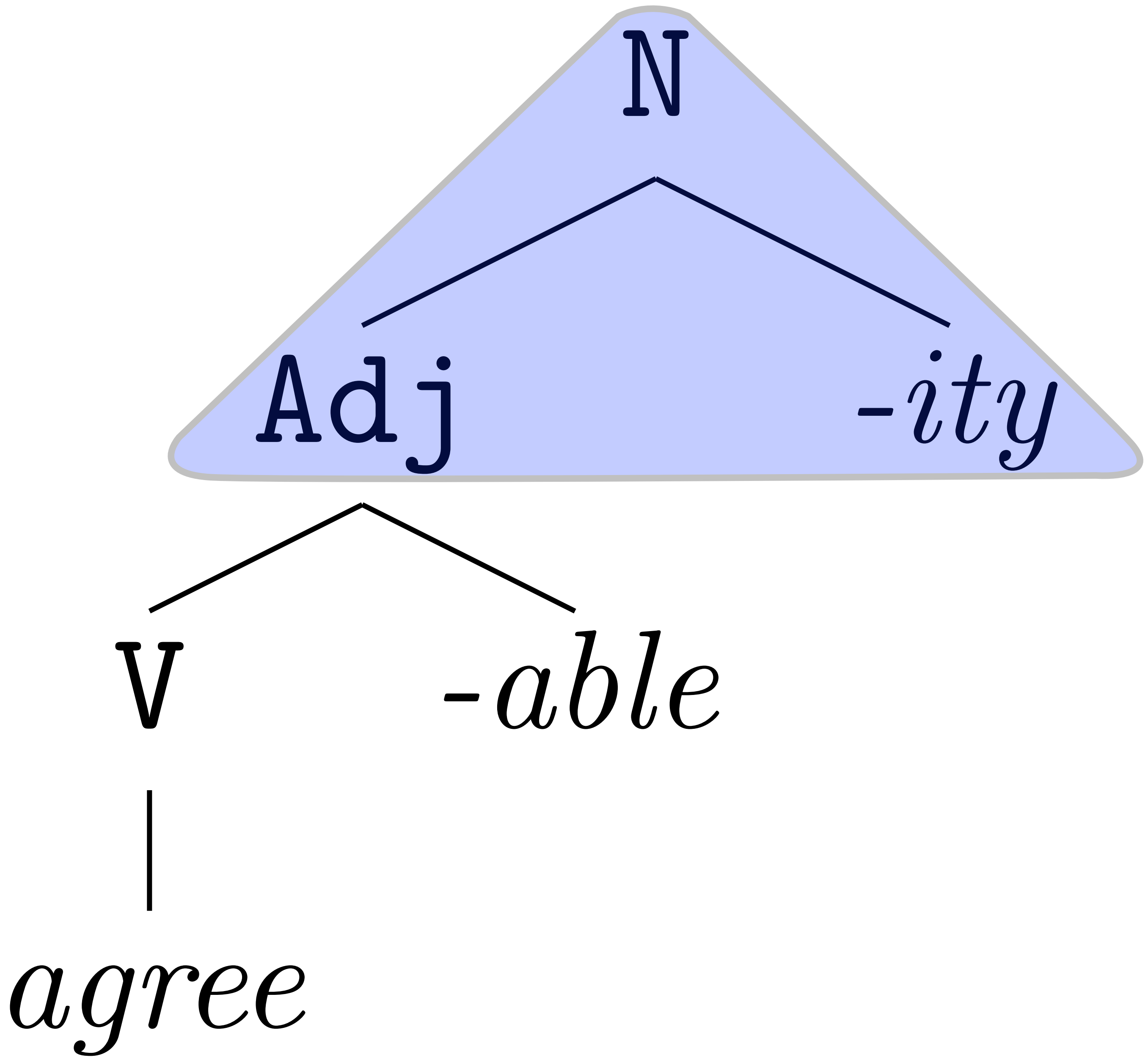


Computation per Expression

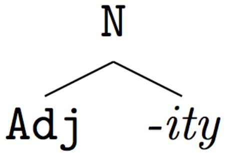


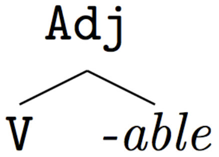
Computation per Expression

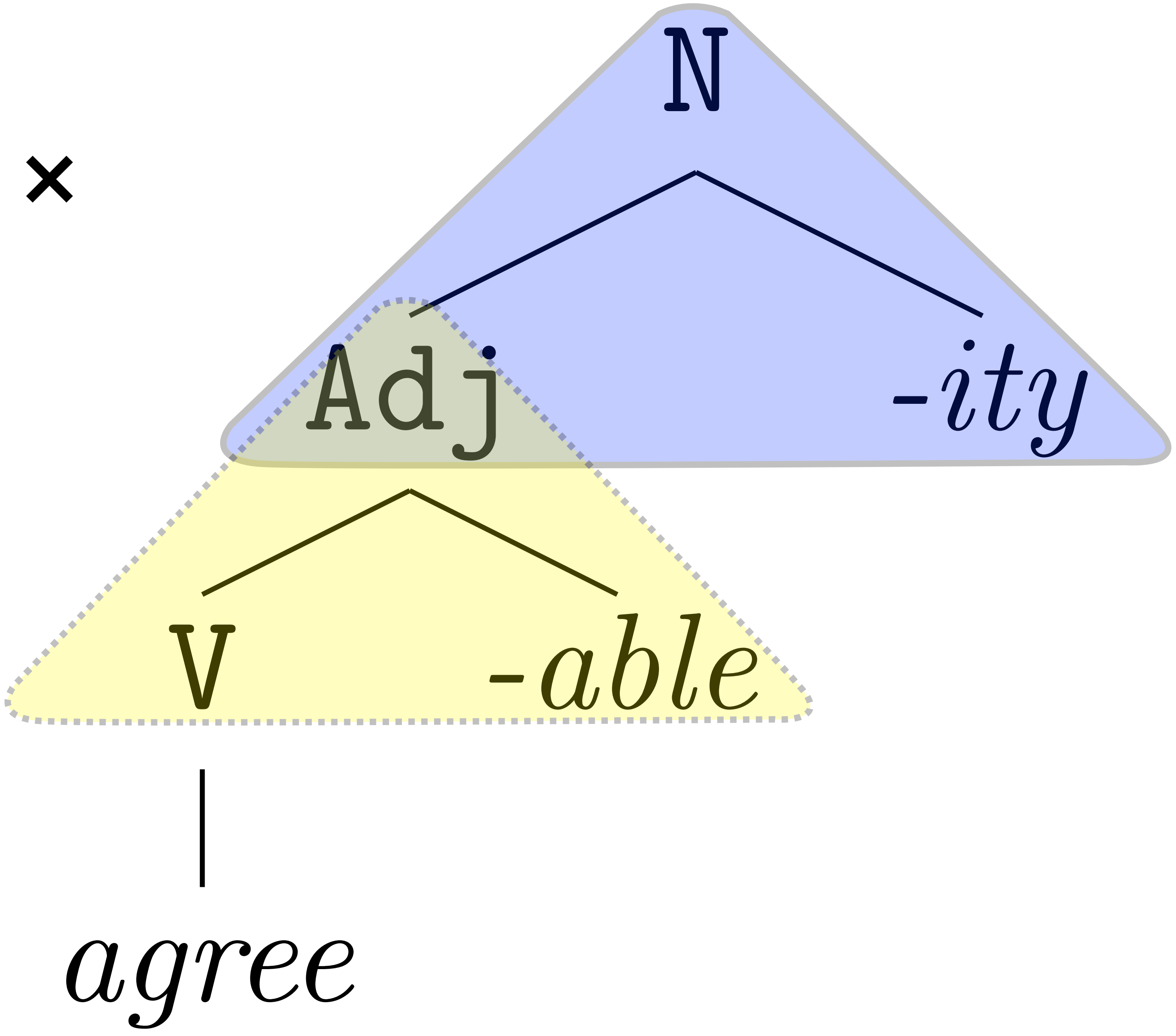
P()



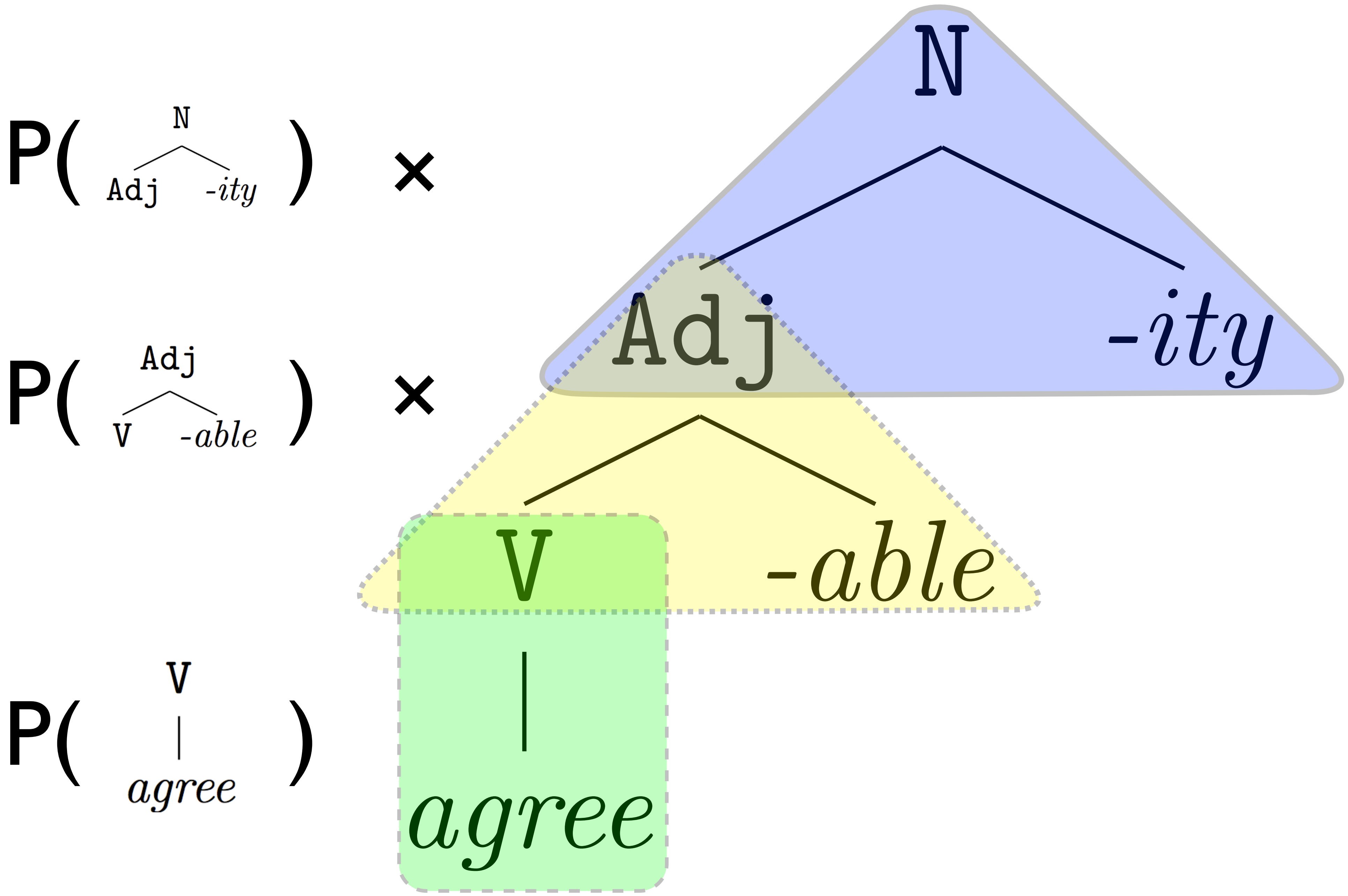
Computation per Expression

P() ×

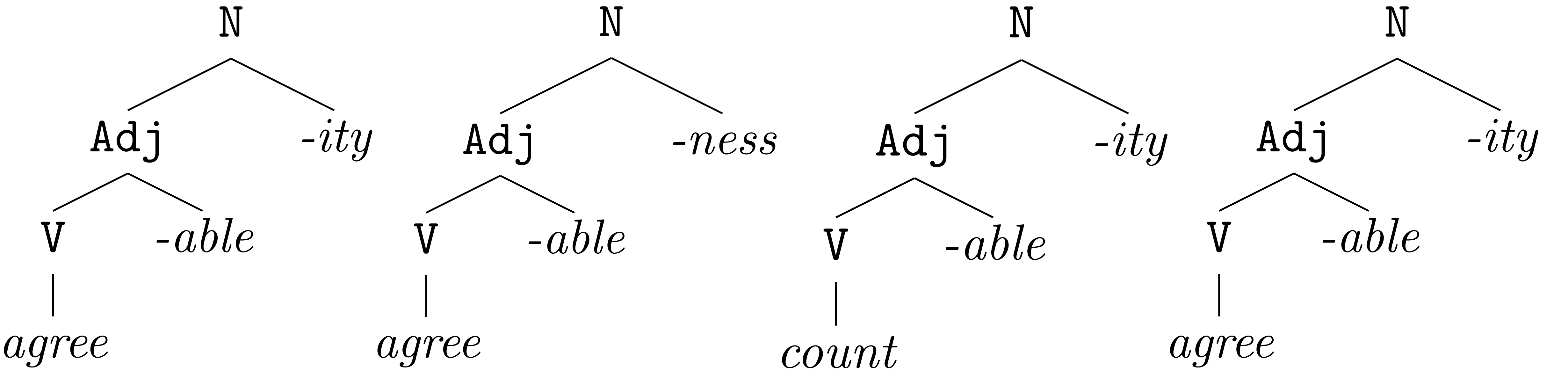
P()



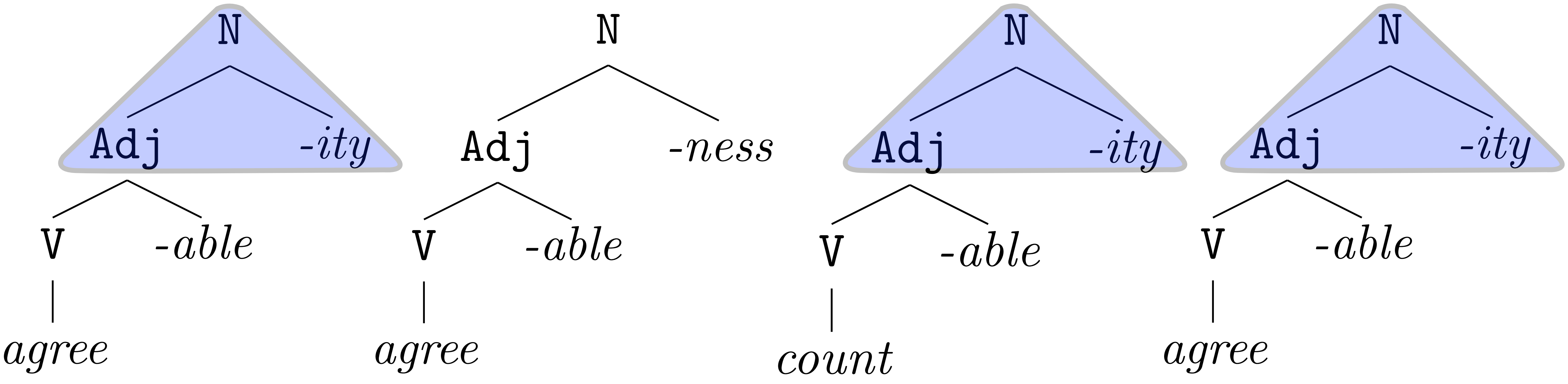
Computation per Expression



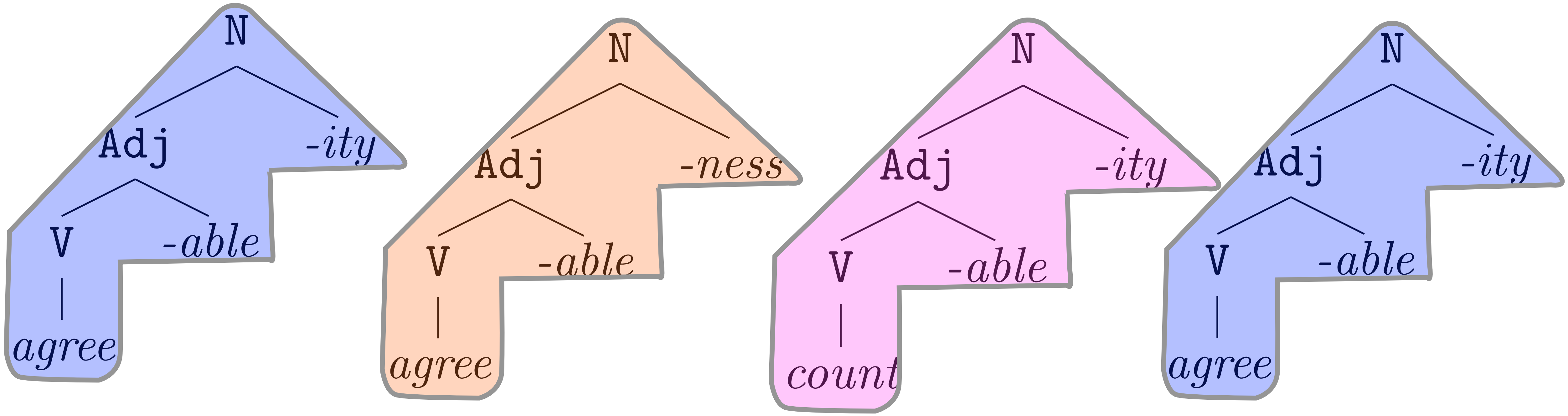
Sharing Across Expressions



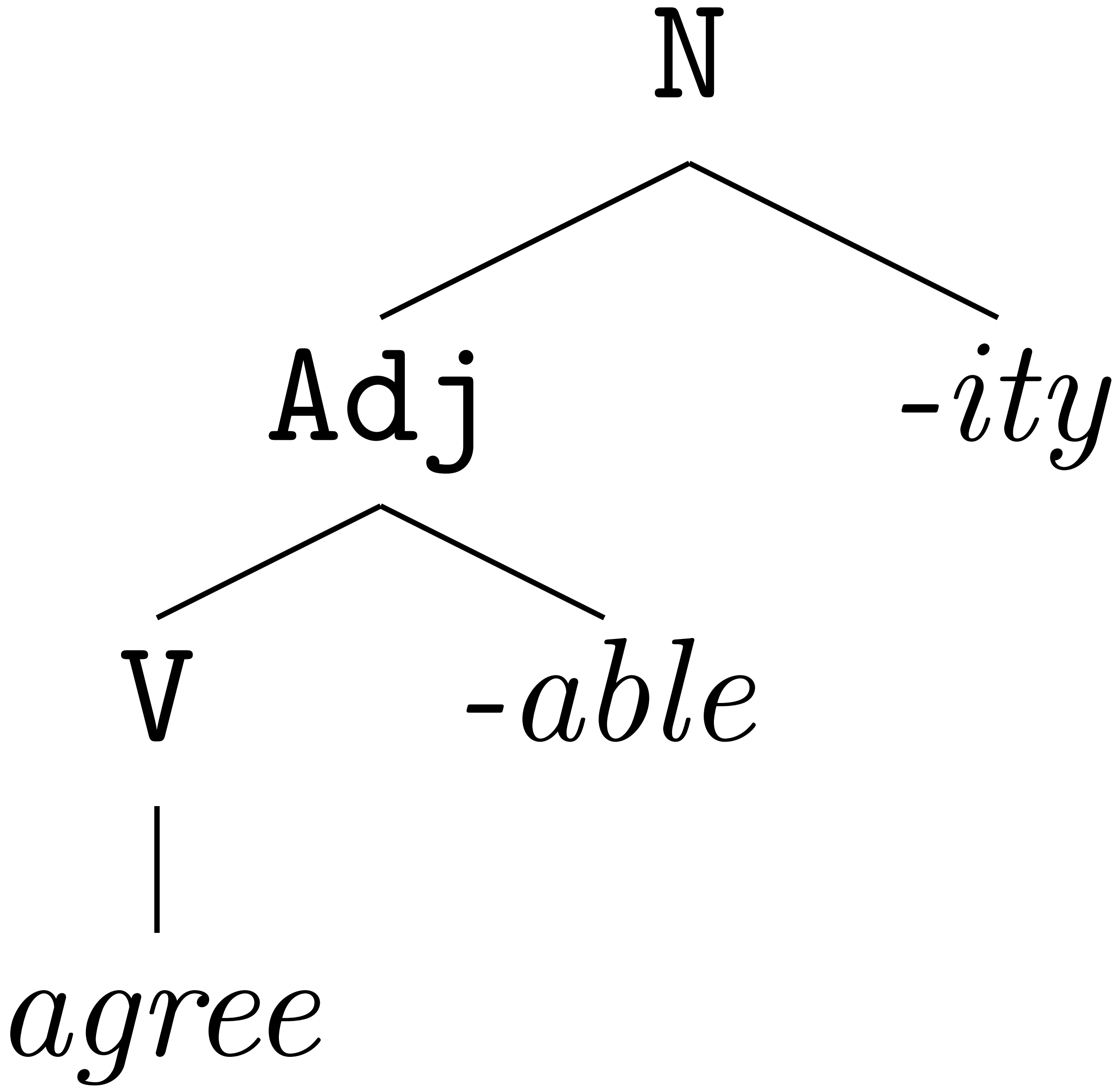
Sharing Across Expressions



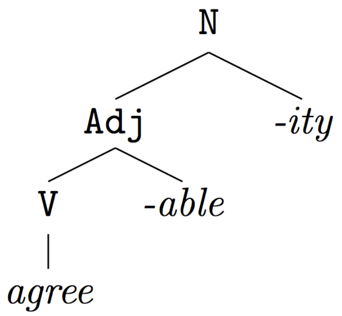
Storage of Maximal, Specific Structures

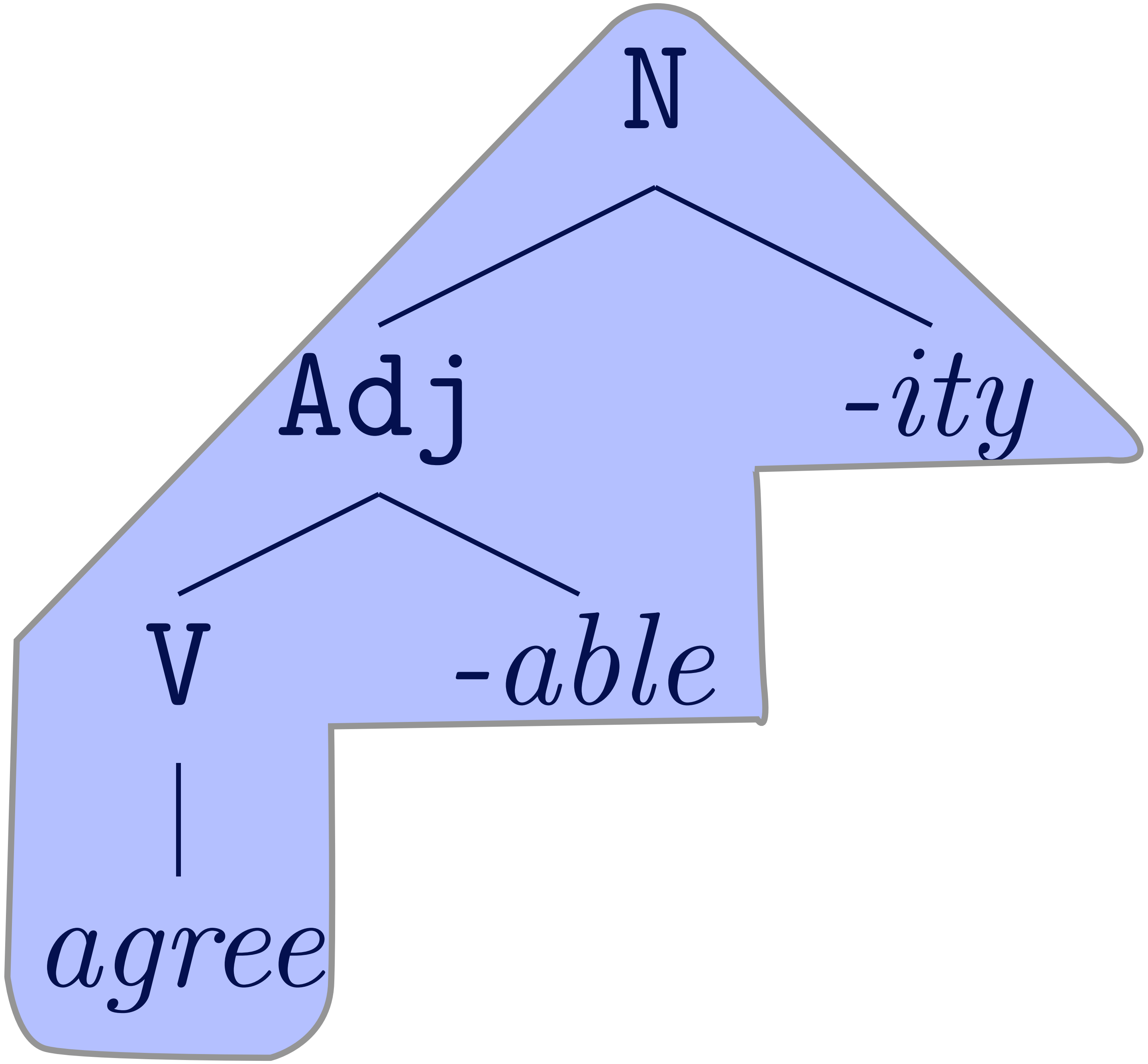


Computation per Expression

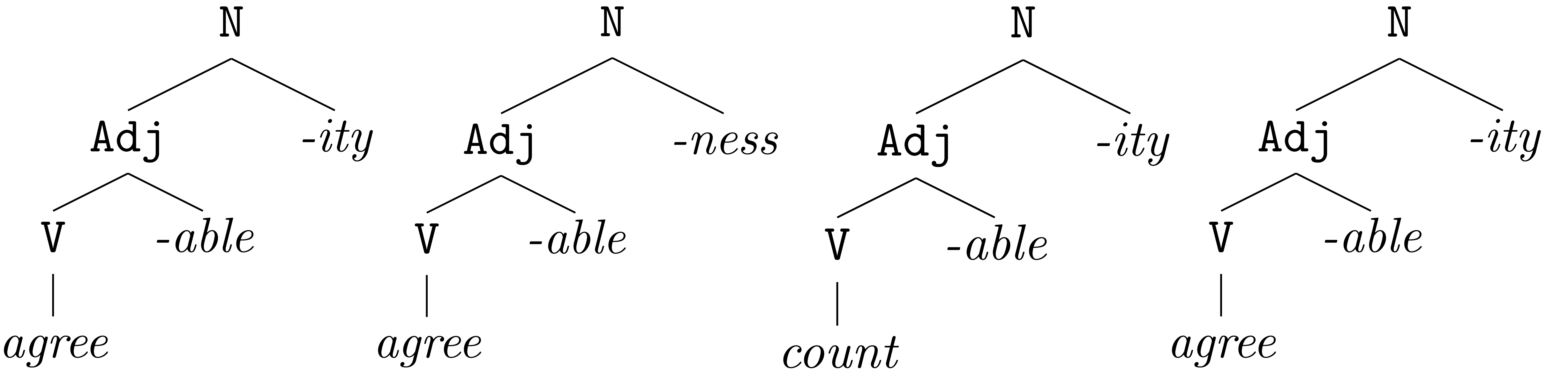


Computation per Expression

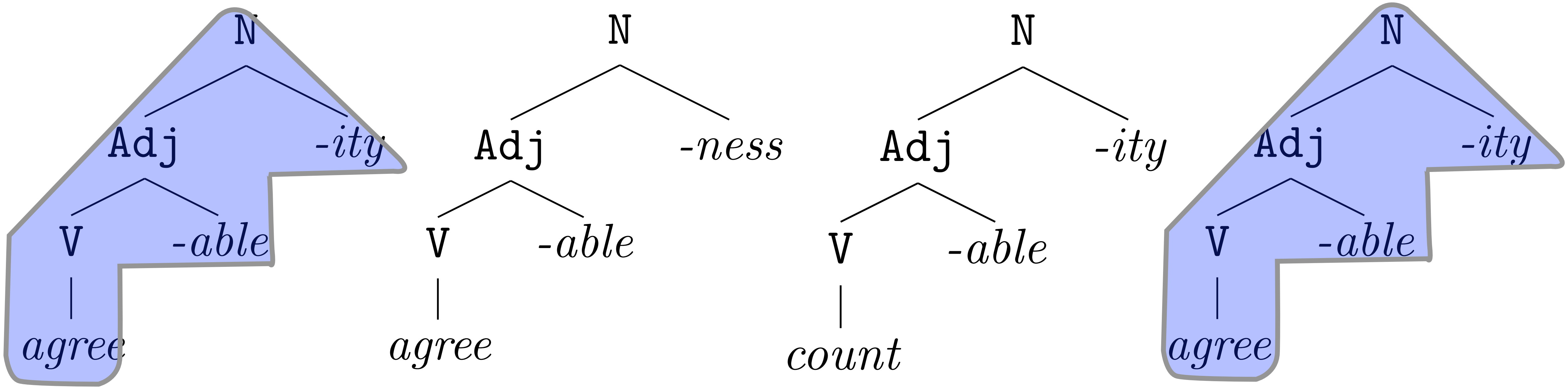
P()



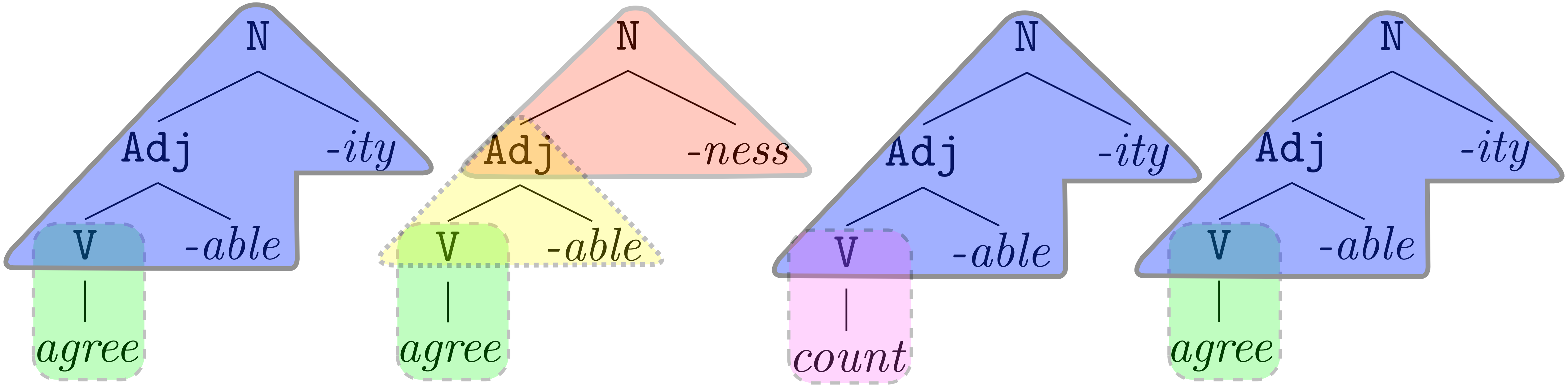
Sharing Across Expressions



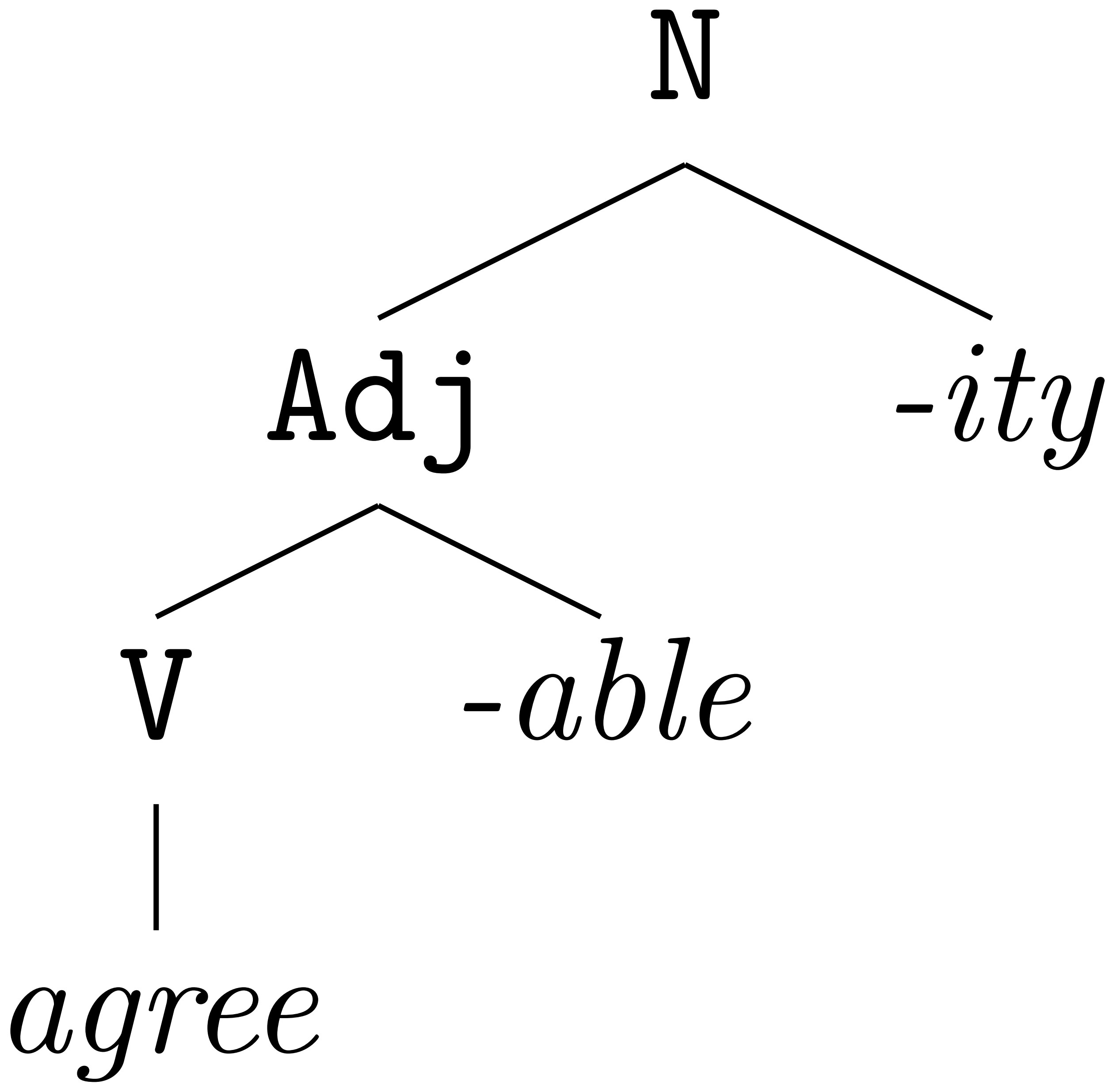
Sharing Across Expressions



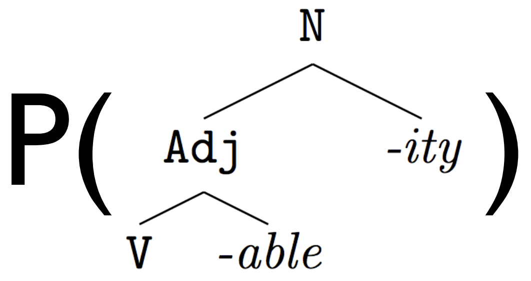
Storage of Intermediate Structures

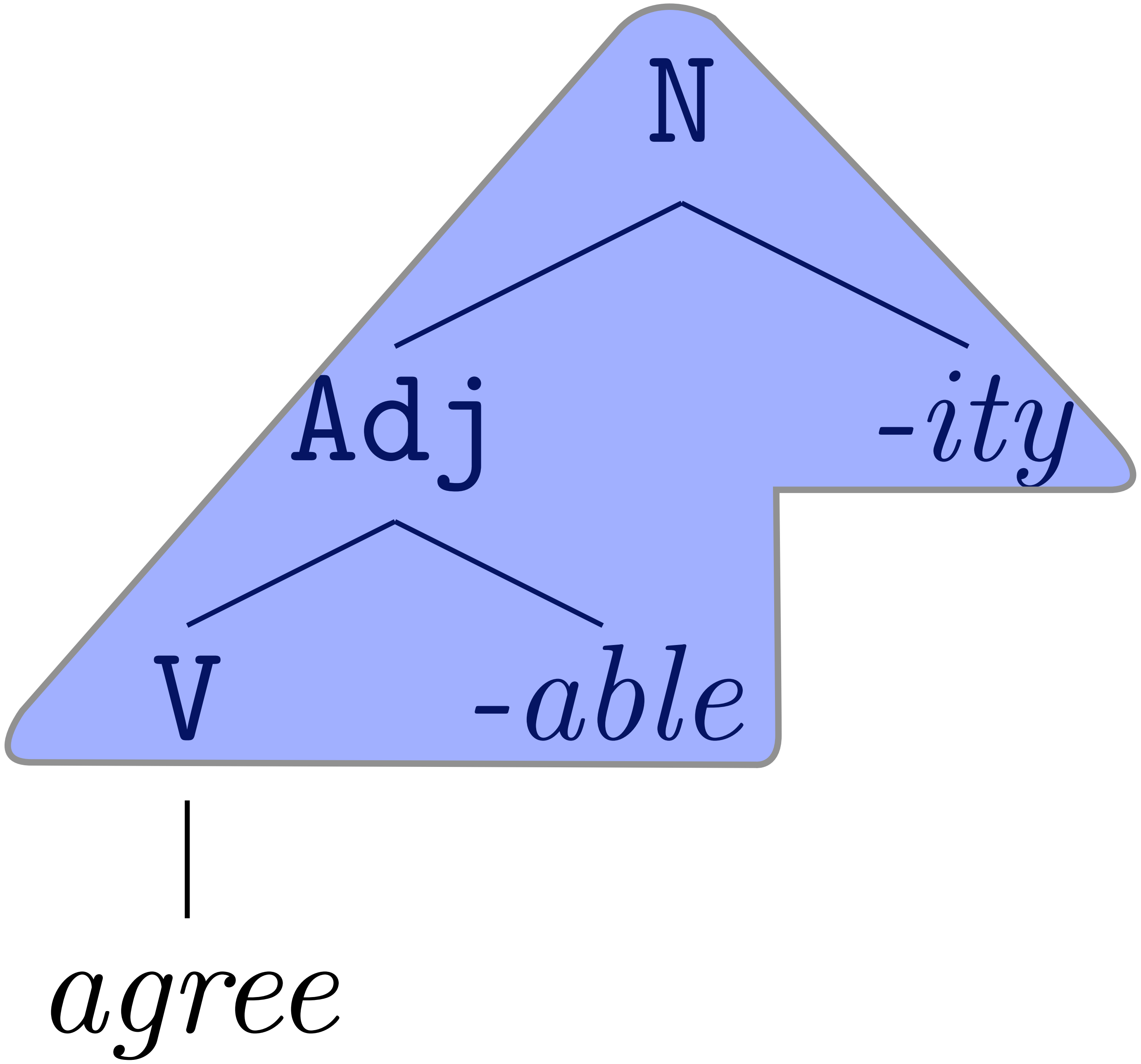


Computation per Expression

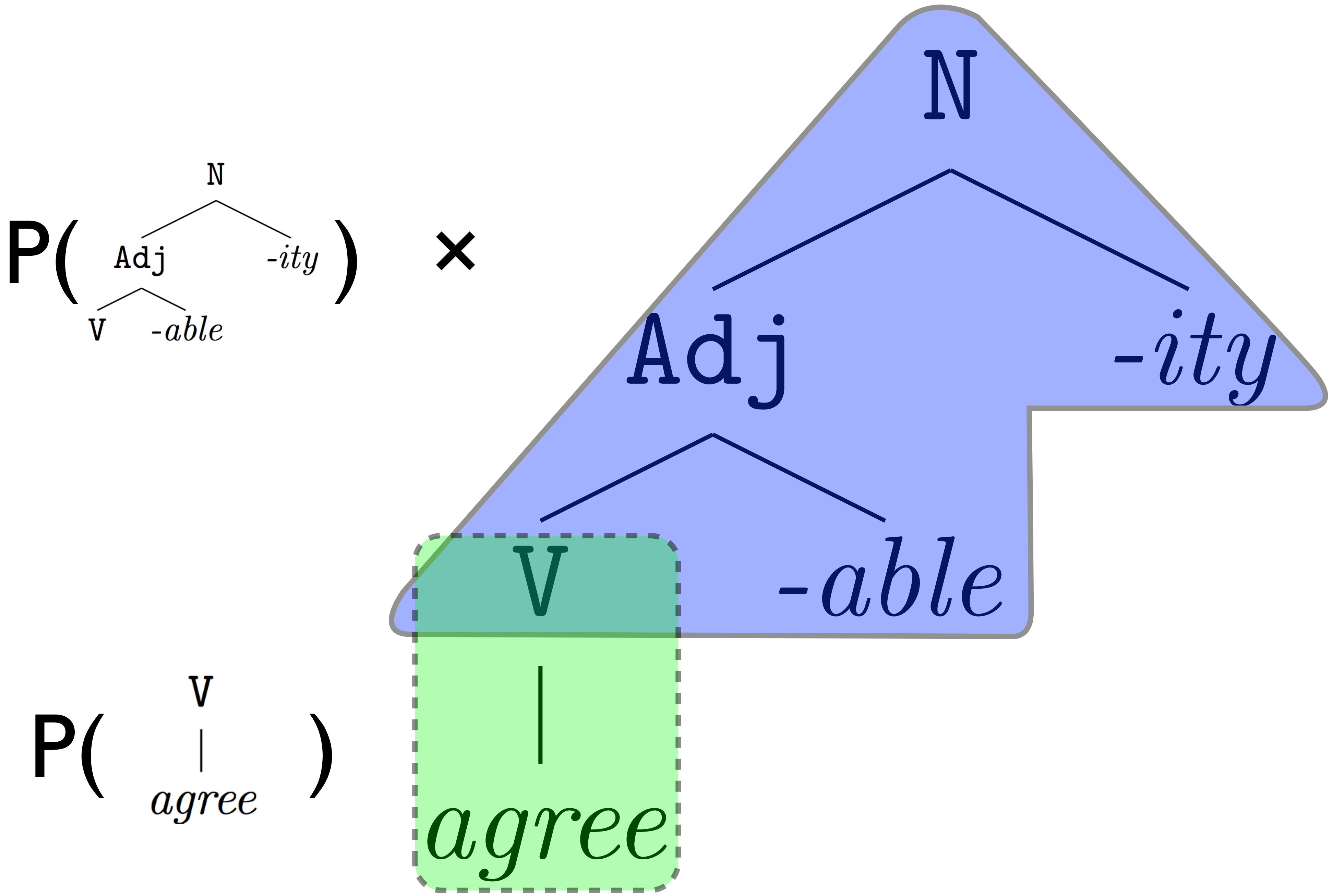


Computation per Expression

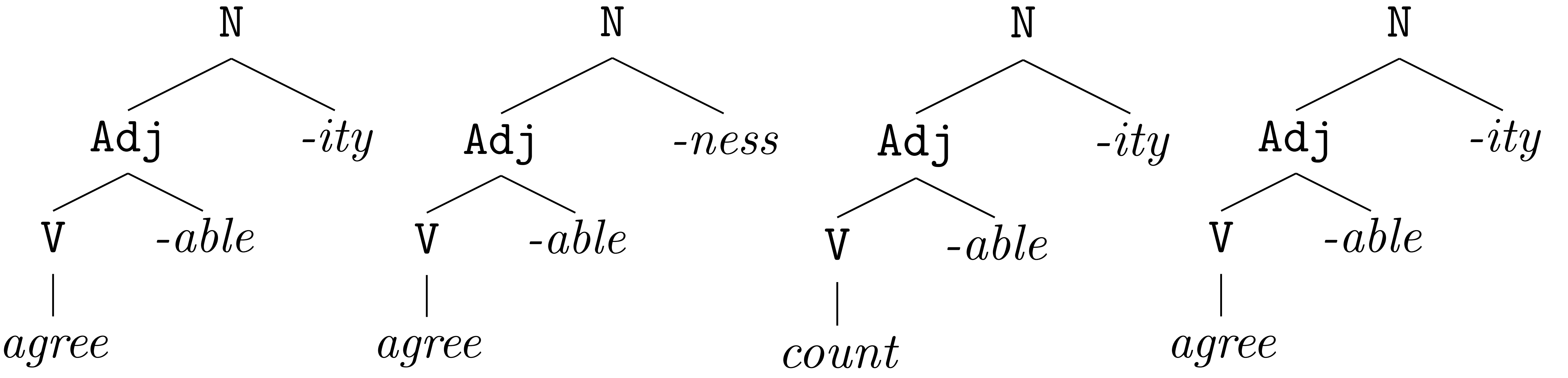




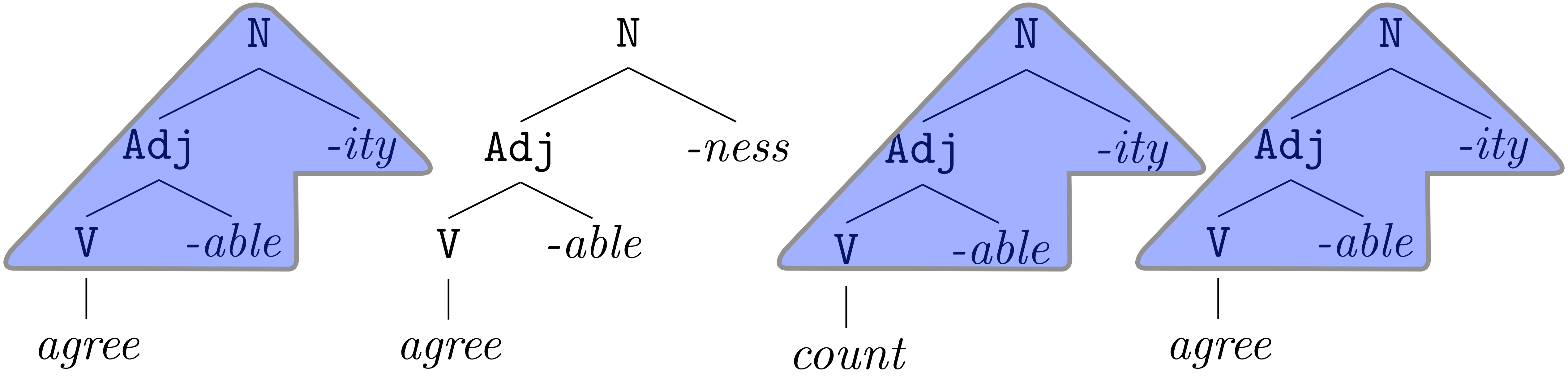
Computation per Expression



Sharing Across Expressions



Sharing Across Expressions



Remarks on Inference Tradeoff

- Nothing fancy here.
- The two simplicity biases are just *Bayesian* **prior** and **likelihood** applied to computation and storage problem.
- **Lexicon code length** and **data code length** given lexicon in (two part) *MDL.*
- Can be connected with many other frameworks.

Inference as Conditioning

- Inference Process: Probabilistic Conditioning.
- Define joint model.

```
P(Data, Fragments) =
     P(Data | Fragments) * P(Fragments)
```

Inference as Conditioning

- Inference Process
- Define joint model.

Likelihood

(derivation probabilities)

```
P(Data, Fragments) =
    P(Data | Fragments) * P(Fragments)
```

Inference as Conditioning

- Inf ditioning.
- Defi

Prior
(lexicon probabilities)

```
P(Data, Fragments) =

          P(Data | Fragments) * P(Fragments)
```

Inference as Conditioning

- Inference Process: Probabilistic Conditioning.
- Condition on particular dataset.

```
P(Fragments | Data) ∝
    P(Data | Fragments) * P(Fragments)
```

Probabilistic Conditioning

- Intuition: two-step algorithm.

1. Throw away lexicons not consistent with the data.
2. Renormalize remaining lexicons so that they sum to one.

- Maximally conservative: **Relative** beliefs are always conserved.

The Mathematical Model: *Fragment Grammars*

- Generalization of *Adaptor Grammars* (Johnson et al., 2007).
    - Allows storing of partial trees.
- Framework first proposed in MDL setting by De Marcken, 1996.
- Related to work on probabilistic tree-substitution grammars (e.g., Bod, 2003; Cohn, 2010; Goodman, 2003; Zuidema, 2007; Post, 2013).

Talk Outline

1. Introduction to productivity and reuse with Fragment Grammars (with Noah Goodman).
2. Case Studies on Productivity and Competition.

Case Studies

- Other approaches to productivity and reuse.

1. What distributions signal productivity?
2. How is competition resolved?
3. Multi-way competition.

Case Studies

- Other approaches to productivity and reuse.

1. What distributions signal productivity?
2. How is competition resolved?
3. Multi-way competition.

Four Strategies for Productivity and Reuse

Four Strategies for Productivity and Reuse

- 5 Formal Models

Four Strategies for Productivity and Reuse

- 5 Formal Models
- Capture historical proposals from the literature.

Four Strategies for Productivity and Reuse

- 5 Formal Models
- Capture historical proposals from the literature.
- Minimally different.

Four Strategies for Productivity and Reuse

- 5 Formal Models
- Capture historical proposals from the literature.
- Minimally different.
    - Same inputs, same underlying space of representations.

Four Strategies for Productivity and Reuse

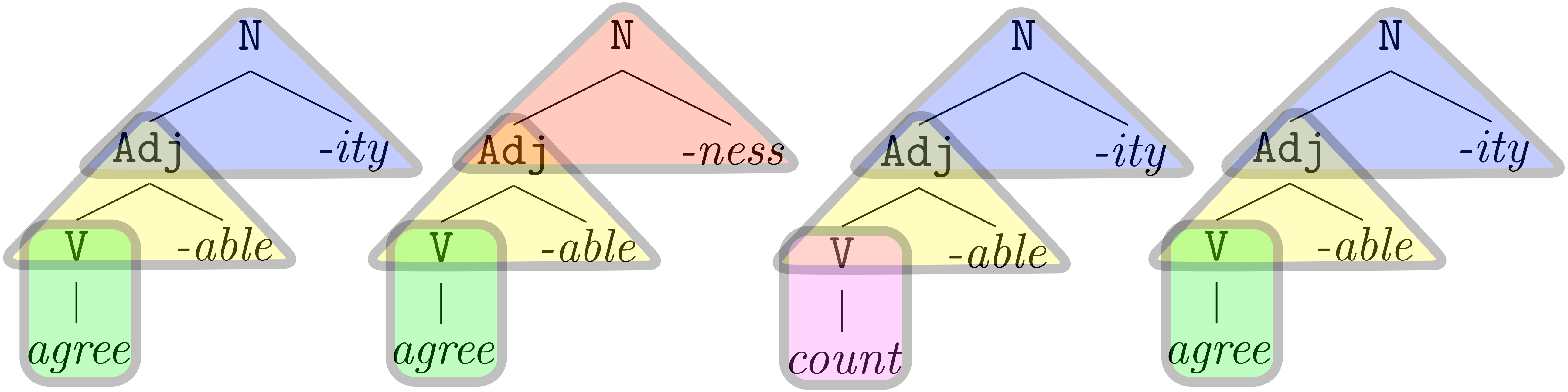
- 5 Formal Models
- Capture historical proposals from the literature.
- Minimally different.
  - Same inputs, same underlying space of representations.
- State-of-the-art probabilistic models.

Full-Parsing
(FP)

Full-Parsing

(MAP *Multinomial-Dirichlet Context-Free Grammars*)

- All generalizations are productive.
- Minimal abstract units.
- Johnson, et al. 2007a
- Estimated on token frequency.

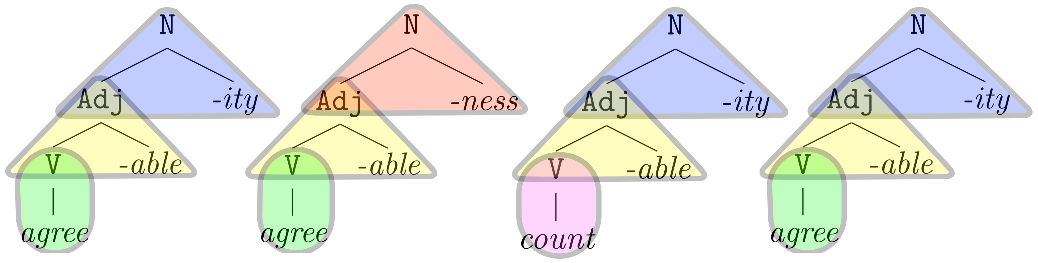


Full-Listing

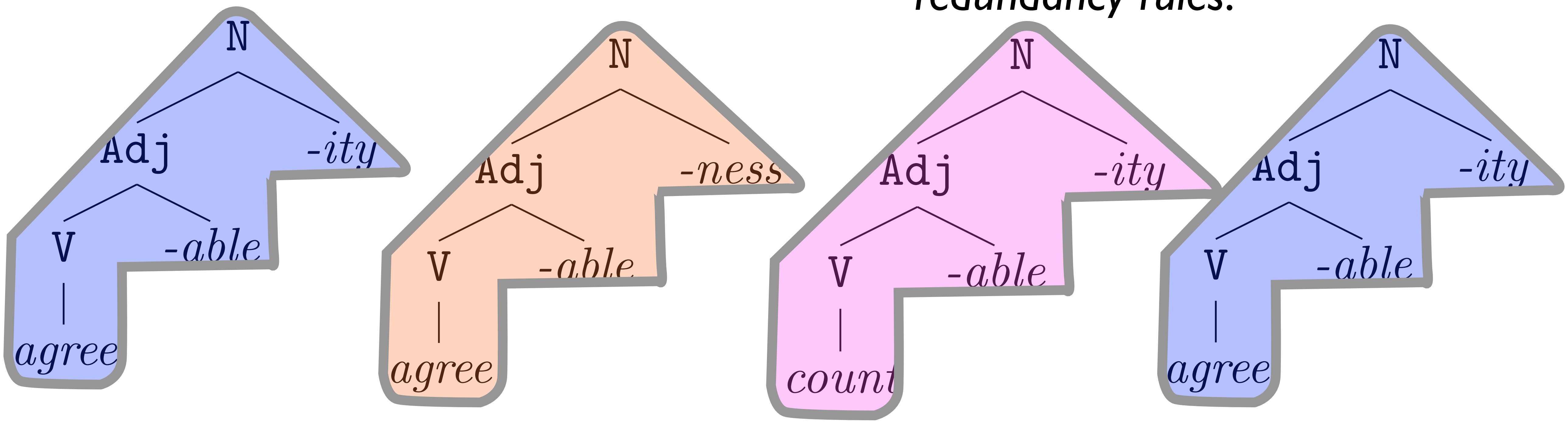
(MAP *All-Adapted Adaptor Grammars*)

- Store whole form after *first* use (recursively).
- Maximally specific units.
- Johnson, et al. 2007
- Base system estimated on type frequencies.
- Formalization of classical *lexical redundancy rules.*

Full-Parsing
(FP)



Full-Listing
(FL)

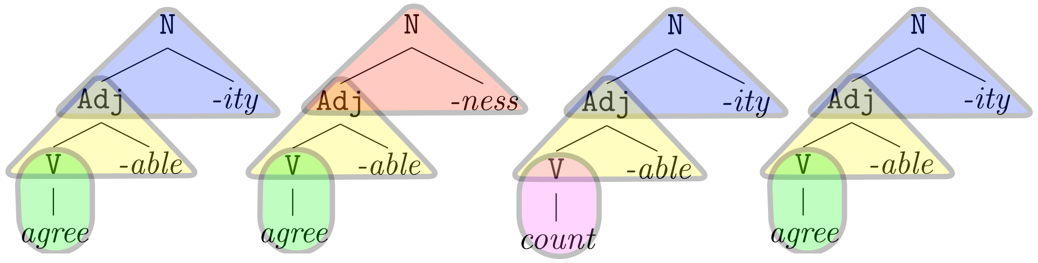


Exemplar-Based

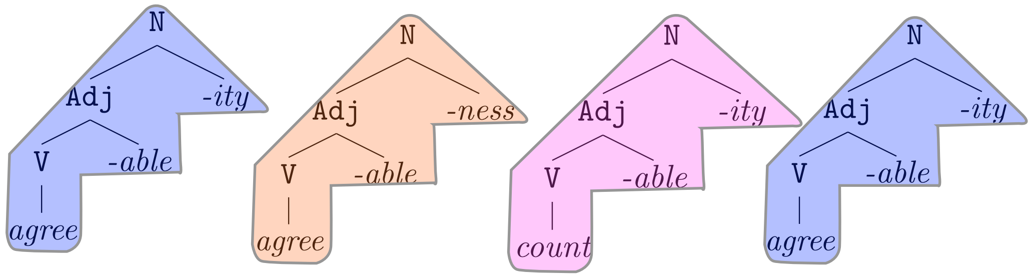
(*Data-Oriented Parsing*)

- Store *all* generalizations consistent with input.
- Two Formalization: *Data-Oriented Parsing 1* (DOP1; Bod, 1998), *Data-Oriented Parsing: Equal-Node Estimator* (ENDOP; Goodman, 2003).
- Argued to be exemplar model of syntax.

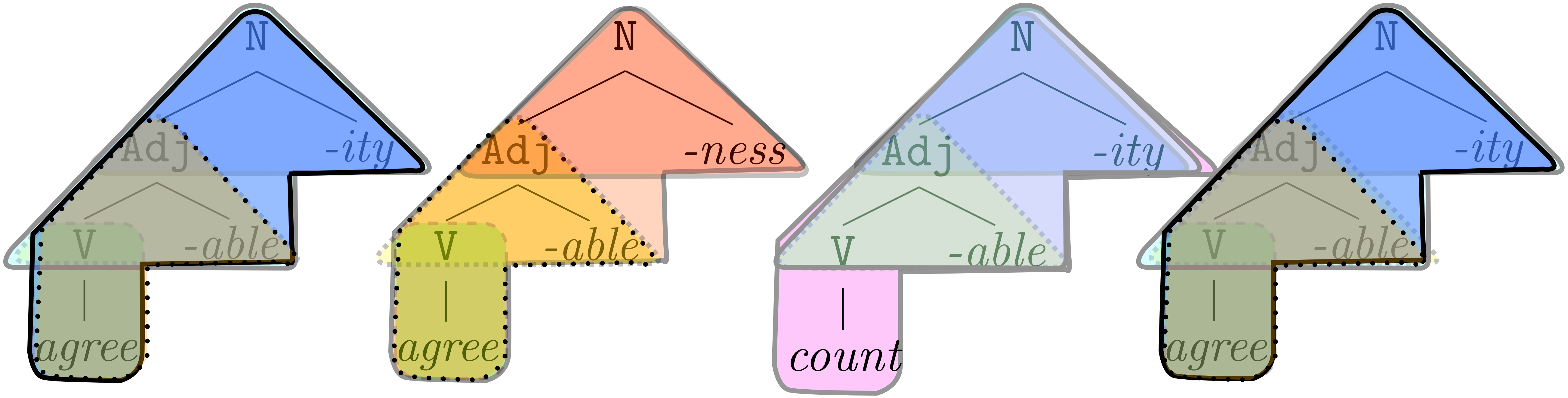
Full-Parsing (FP)



Full-Listing (FL)



Exemplar-Based (EB)



Inference-Based

(*Fragment Grammars*)

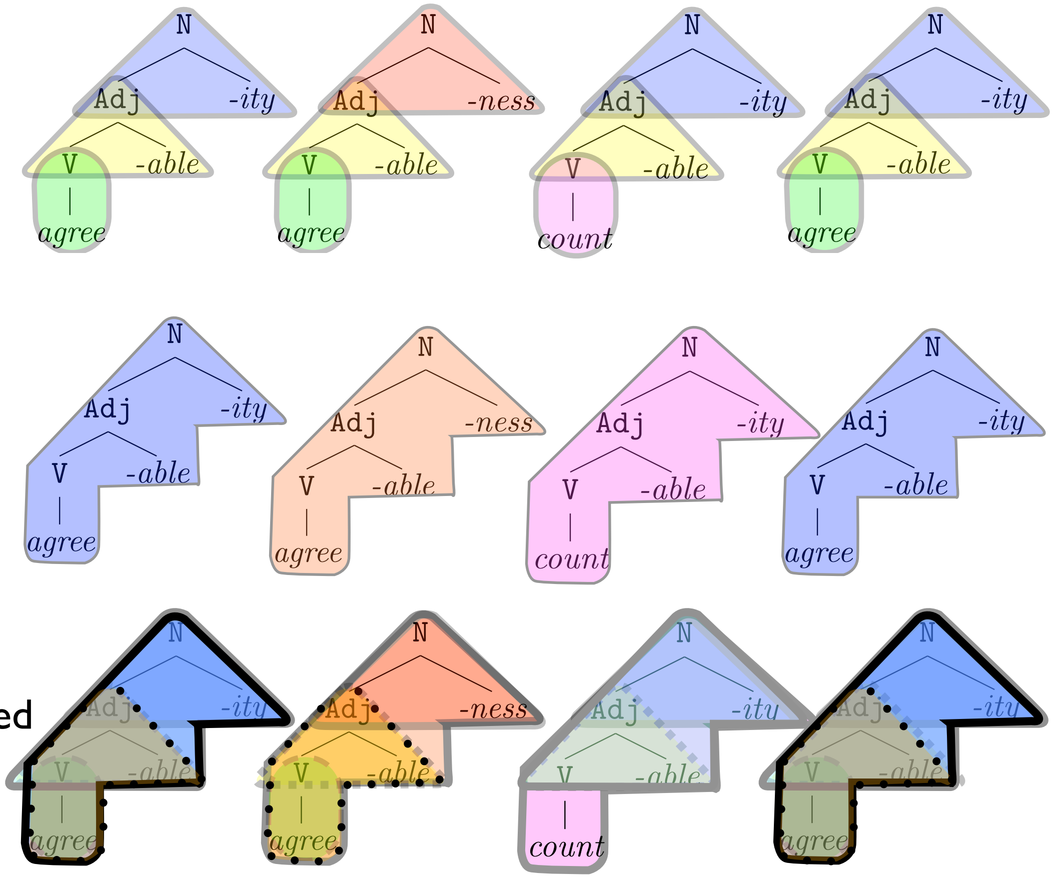
- Store set of subcomputations which *best* explains the data.
- Formalization: *Fragment Grammars* (O'Donnell, et al. 2009)
- Inference depends on distribution of tokens over types.
- Only model which infers variables.

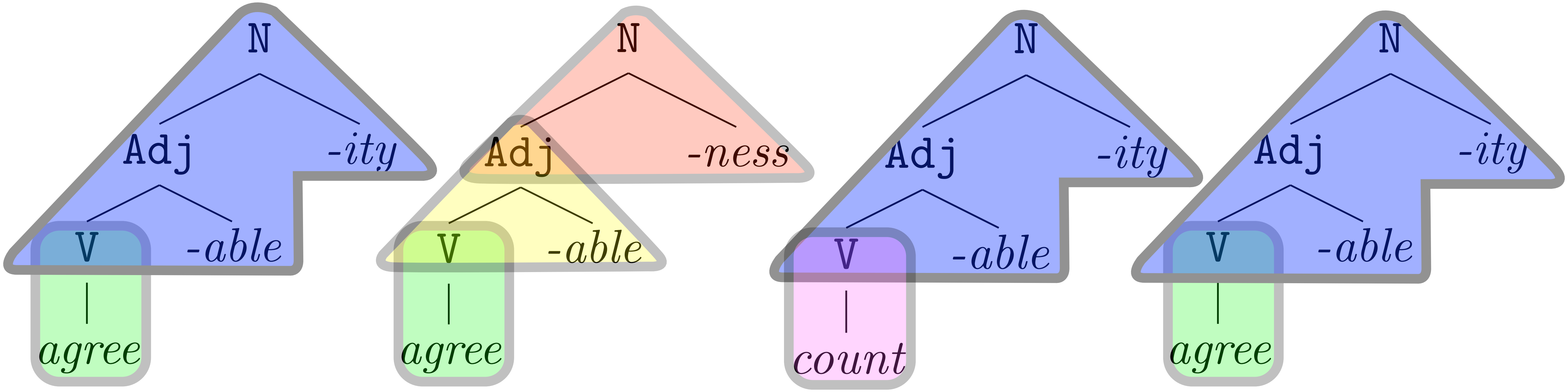
Full-Parsing (FP)

Full-Listing (FL)

Exemplar-Based (EB)

Inference-Based (IB)





Empirical Domains

	Past Tense (Inflectional)	Derivational Morphology
Productive	+ed *(walked)*	+ness *(goodness)*
Context-Dependent	I → æ *(sang)*	+ity *(ability)*
Unproductive	suppletion *(go/went)*	+th *(width)*

Case Studies

- Other approaches to productivity and reuse.

1. What distributions signal productivity?
2. How is competition resolved?
3. Multi-way competition.

Empirical Evaluations

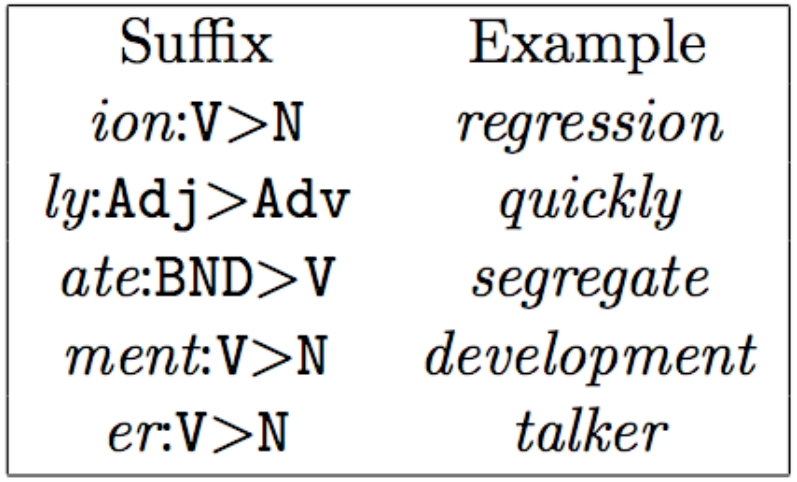
	Past Tense	Derivational Morphology
Productive	+ed *(walked)*	+ness *(goodness)*
Context-Dependent	I → æ *(sang)*	+ity *(ability)*
Unproductive	suppletion *(go/went)*	+th *(width)*

What (Distributional) Cues Signal Productivity?

- Many proposals in the literature:
  - Type frequency.
  - Token frequency (combined with something else, e.g., entropy).
  - Heterogeneity of context (generalized type frequency).

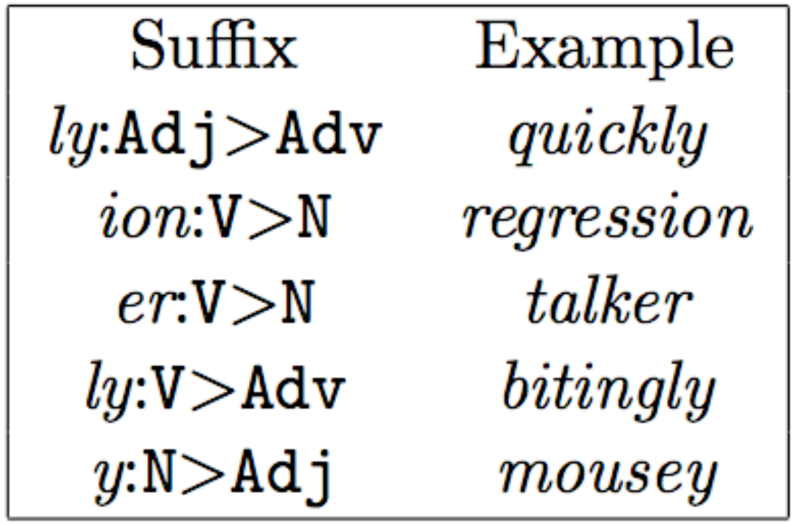
Top 5 Most Productive Suffixes

Full-Parsing (MDPCFG)



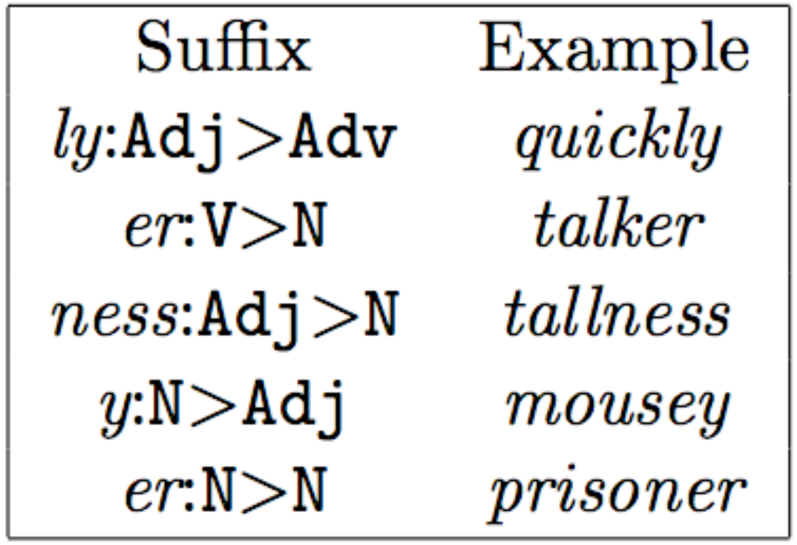
Suffix	Example
ion:V>N	*regression*
ly:Adj>Adv	*quickly*
ate:BND>V	*segregate*
ment:V>N	*development*
er:V>N	*talker*

Full-Listing (MAG)



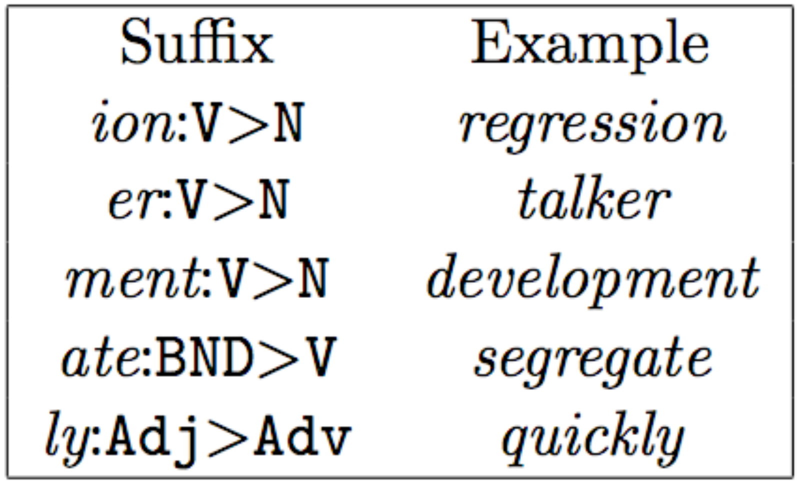
Suffix	Example
ly:Adj>Adv	*quickly*
ion:V>N	*regression*
er:V>N	*talker*
ly:V>Adv	*bitingly*
y:N>Adj	*mousey*

Inference-Based (FG)



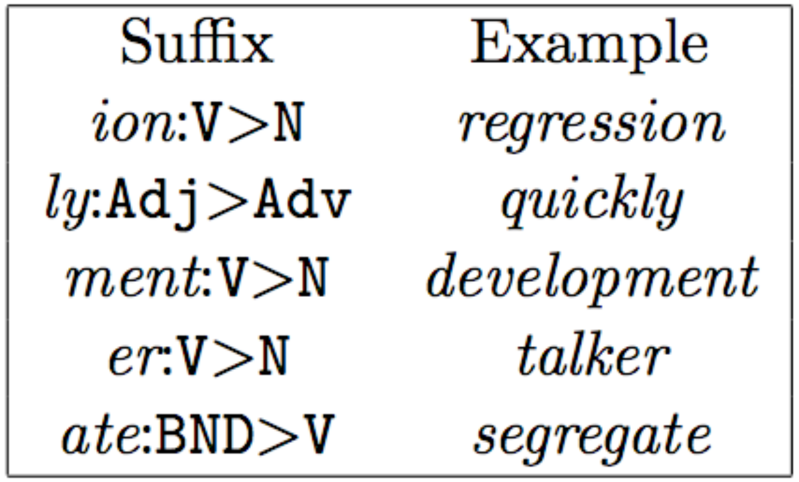
Suffix	Example
ly:Adj>Adv	*quickly*
er:V>N	*talker*
ness:Adj>N	*tallness*
y:N>Adj	*mousey*
er:N>N	*prisoner*

Exemplar (DOP1)



Suffix	Example
ion:V>N	*regression*
er:V>N	*talker*
ment:V>N	*development*
ate:BND>V	*segregate*
ly:Adj>Adv	*quickly*

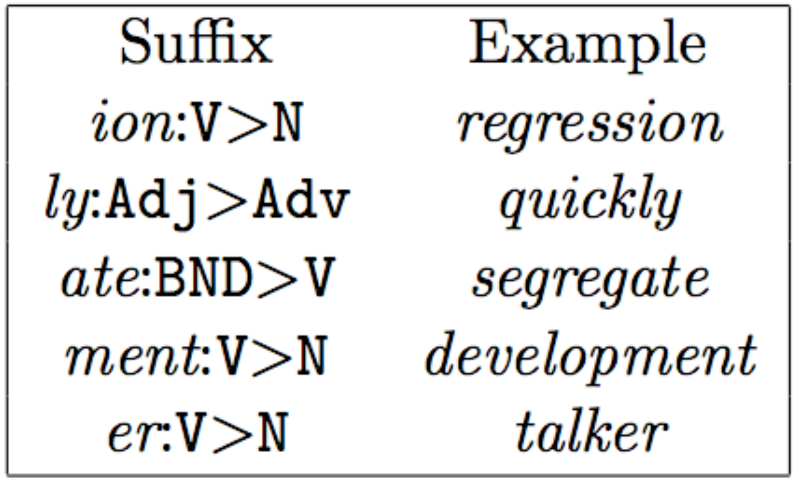
Exemplar (ENDOP)



Suffix	Example
ion:V>N	*regression*
ly:Adj>Adv	*quickly*
ment:V>N	*development*
er:V>N	*talker*
ate:BND>V	*segregate*

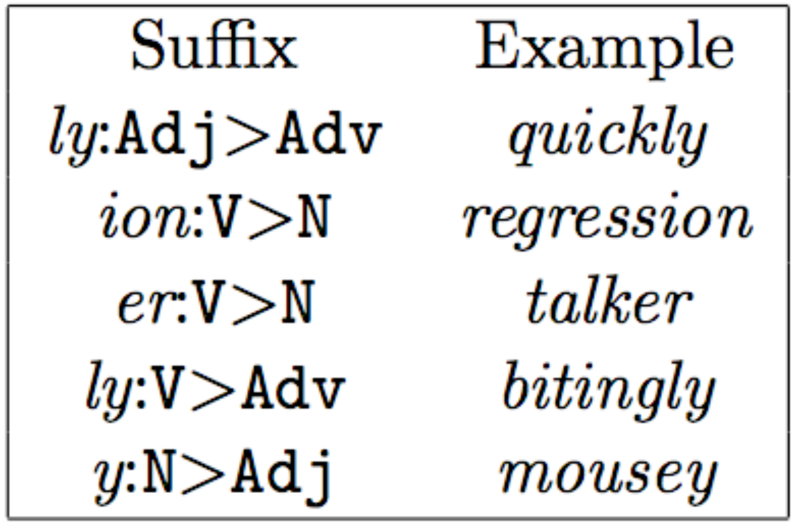
Top 5 Most Productive Suffixes

Full-Parsing (MDPCFG)



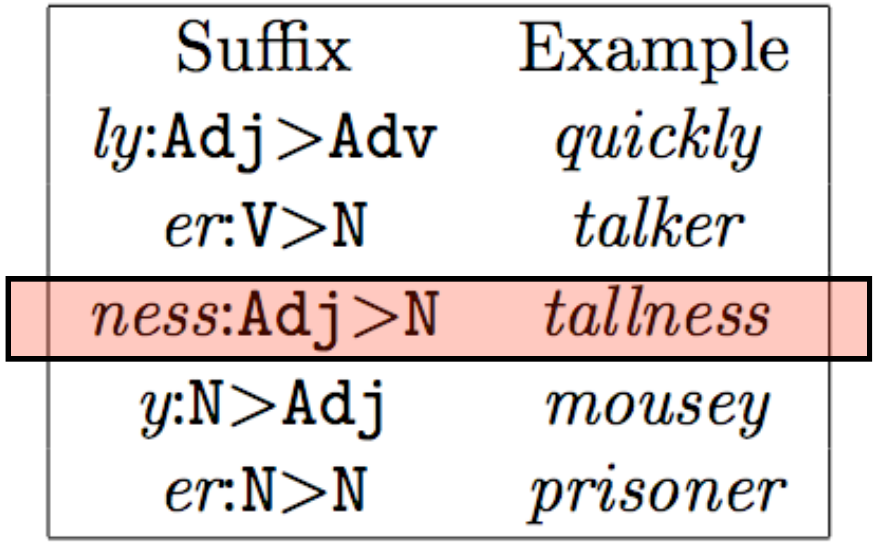
Suffix	Example
ion:V>N	*regression*
ly:Adj>Adv	*quickly*
ate:BND>V	*segregate*
ment:V>N	*development*
er:V>N	*talker*

Full-Listing (MAG)



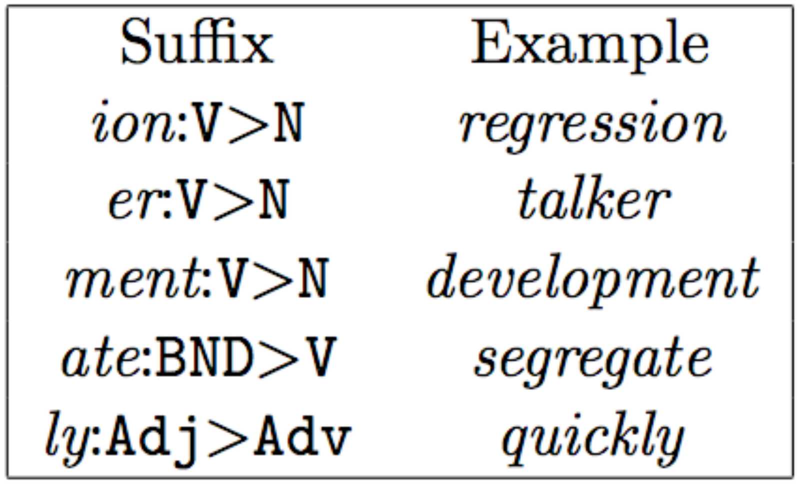
Suffix	Example
ly:Adj>Adv	*quickly*
ion:V>N	*regression*
er:V>N	*talker*
ly:V>Adv	*bitingly*
y:N>Adj	*mousey*

Inference-Based (FG)



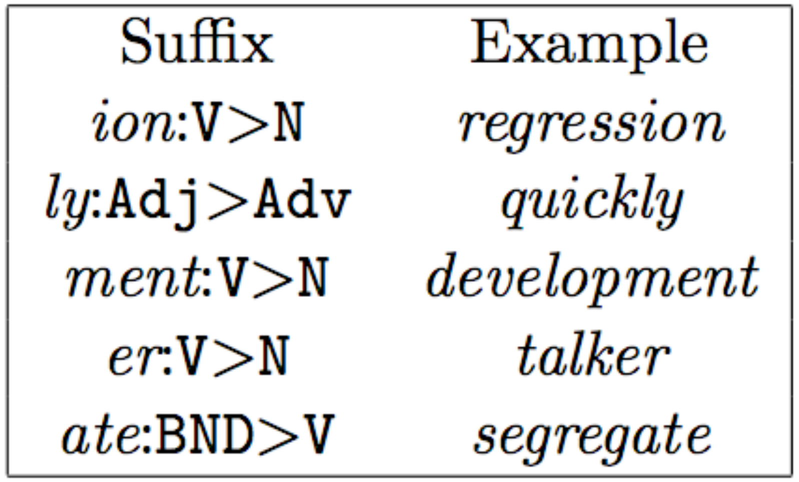
Suffix	Example
ly:Adj>Adv	*quickly*
er:V>N	*talker*
ness:Adj>N	*tallness*
y:N>Adj	*mousey*
er:N>N	*prisoner*

Exemplar (DOP1)



Suffix	Example
ion:V>N	*regression*
er:V>N	*talker*
ment:V>N	*development*
ate:BND>V	*segregate*
ly:Adj>Adv	*quickly*

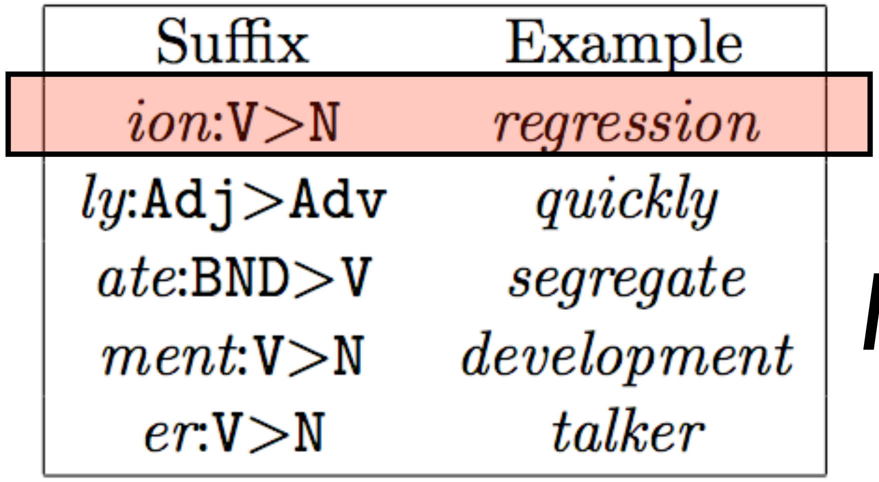
Exemplar (ENDOP)



Suffix	Example
ion:V>N	*regression*
ly:Adj>Adv	*quickly*
ment:V>N	*development*
er:V>N	*talker*
ate:BND>V	*segregate*

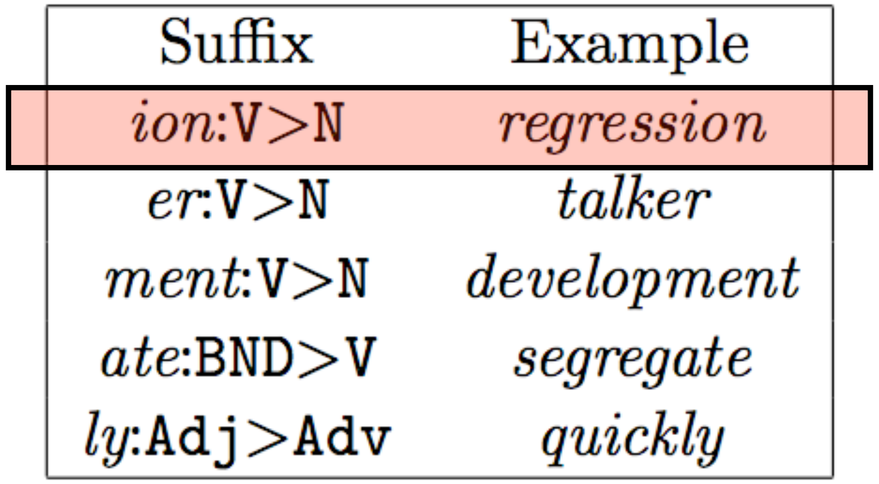
Top 5 Most Productive Suffixes

Full-Parsing (MDPCFG)



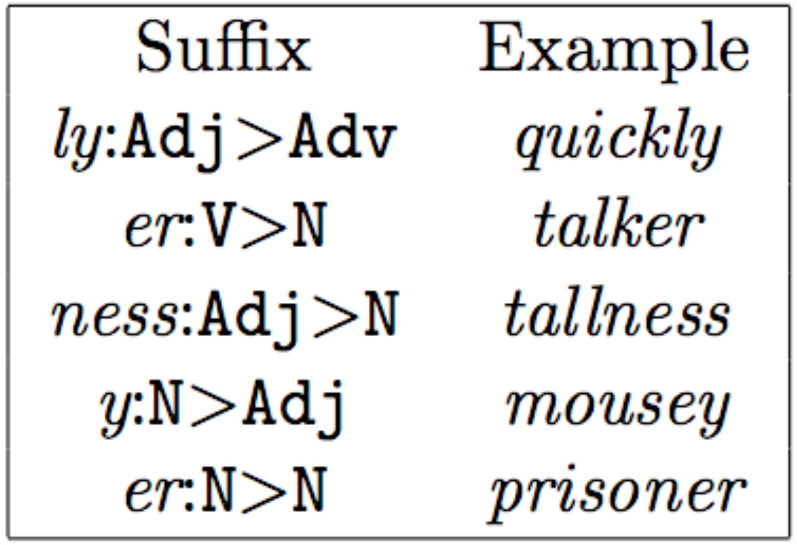
Suffix	Example
ion:V>N	*regression*
ly:Adj>Adv	*quickly*
ate:BND>V	*segregate*
ment:V>N	*development*
er:V>N	*talker*

Exemplar (DOP1)



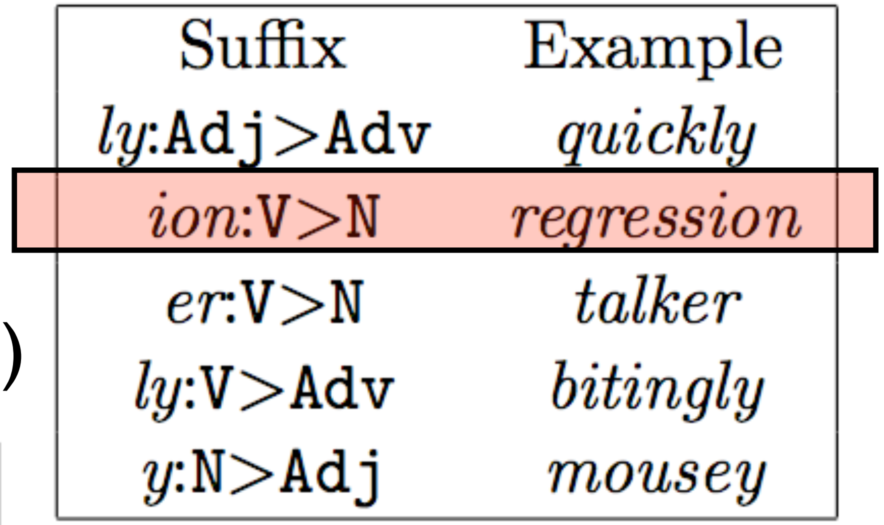
Suffix	Example
ion:V>N	*regression*
er:V>N	*talker*
ment:V>N	*development*
ate:BND>V	*segregate*
ly:Adj>Adv	*quickly*

Inference-Based (FG)



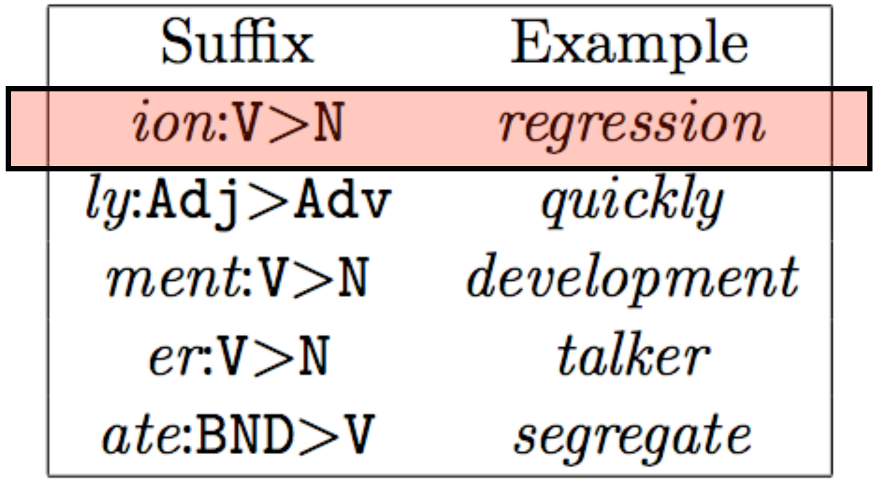
Suffix	Example
ly:Adj>Adv	*quickly*
er:V>N	*talker*
ness:Adj>N	*tallness*
y:N>Adj	*mousey*
er:N>N	*prisoner*

Full-Listing (MAG)



Suffix	Example
ly:Adj>Adv	*quickly*
ion:V>N	*regression*
er:V>N	*talker*
ly:V>Adv	*bitingly*
y:N>Adj	*mousey*

Exemplar (GDMN)



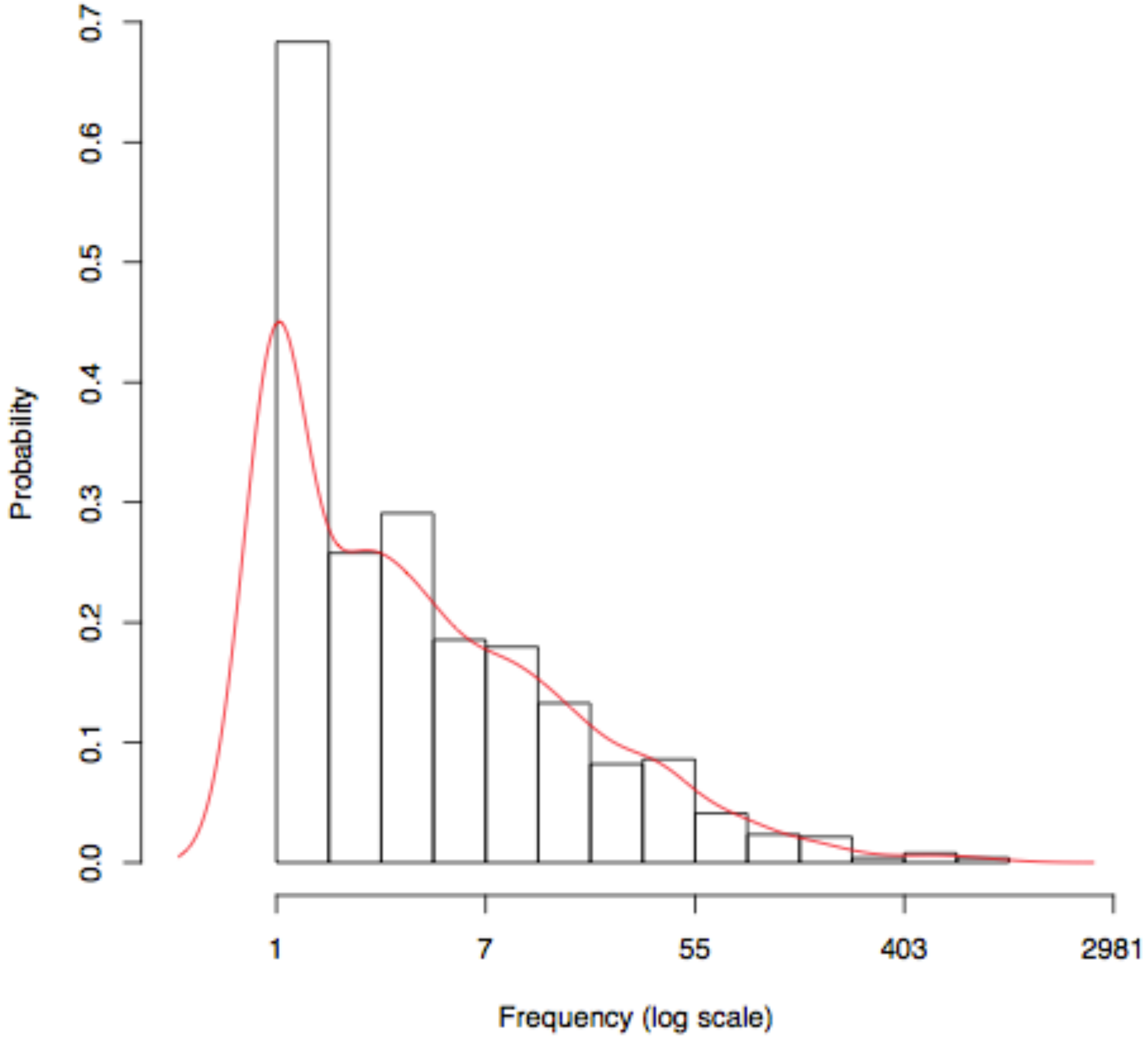
Suffix	Example
ion:V>N	*regression*
ly:Adj>Adv	*quickly*
ment:V>N	*development*
er:V>N	*talker*
ate:BND>V	*segregate*

What Evidences Productivity?

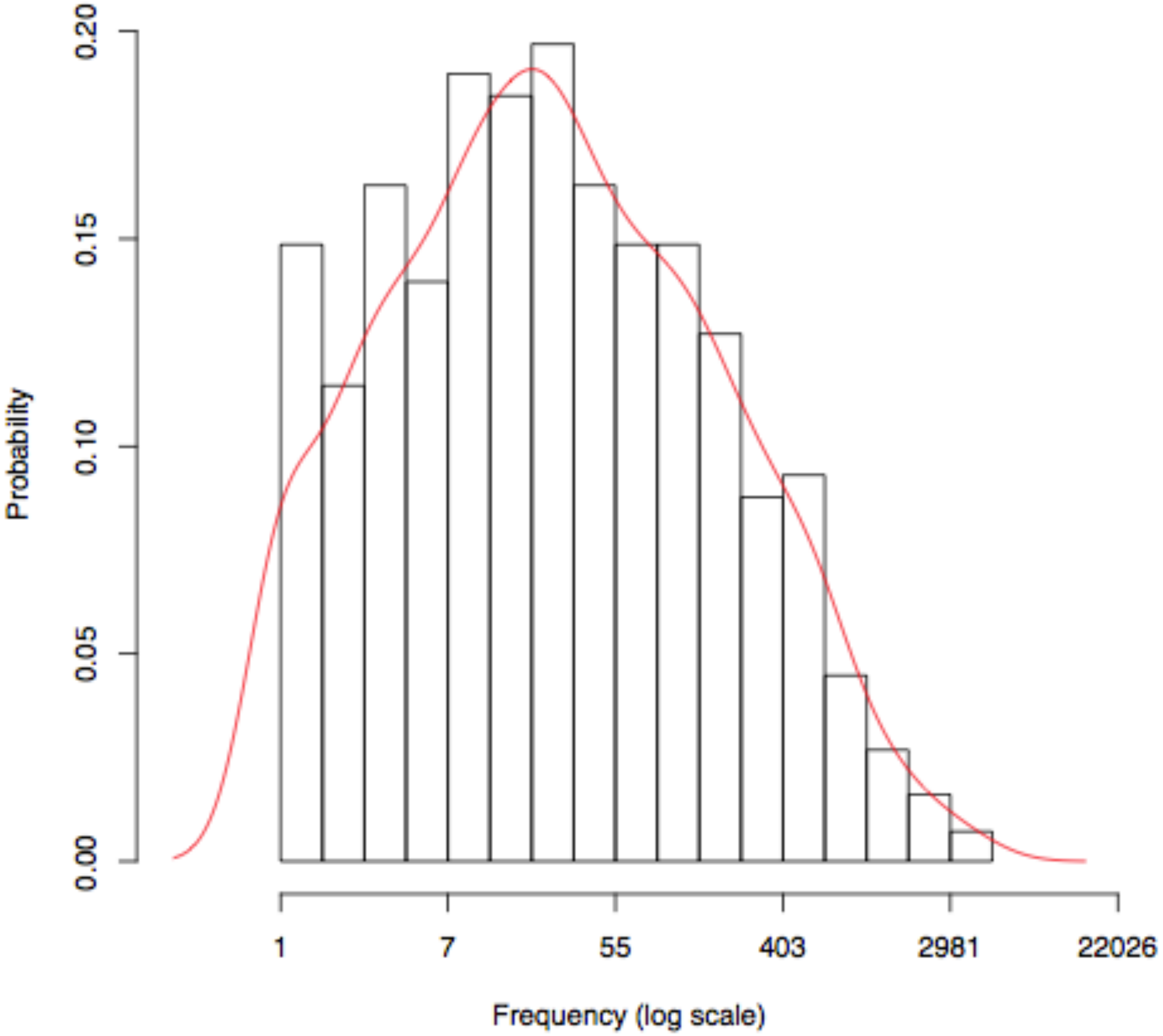
- Crucial evidence of productivity: Use of a lexical item (morpheme, rule, etc.) to generate new forms.
- Distributional consequence: Large proportion of low frequency forms.

What Predicts Productivity?

-ness:`Adj>N`

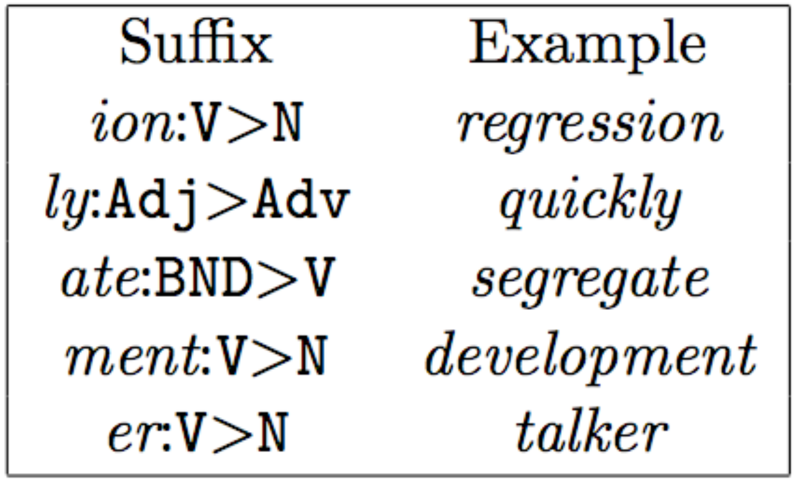


-ion:`V>N`



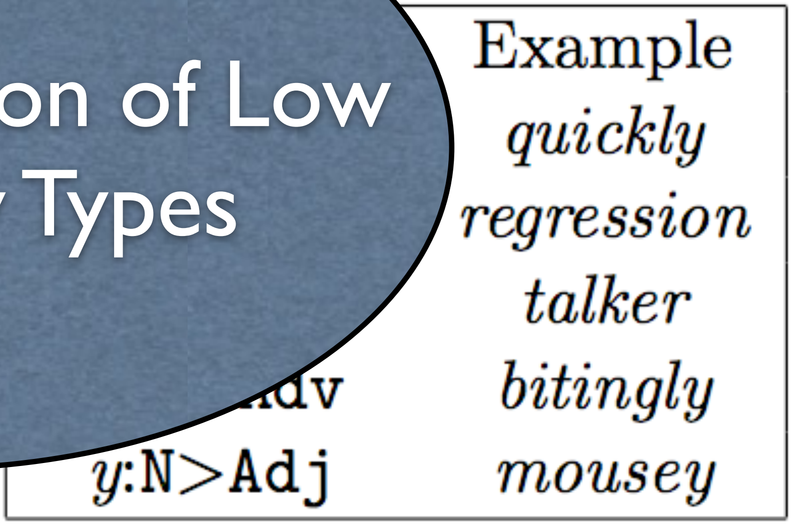
Top 5 Most Productive Suffixes

Full-Parsing (MDPCFG)



Suffix	Example
ion:V>N	*regression*
ly:Adj>Adv	*quickly*
ate:BND>V	*segregate*
ment:V>N	*development*
er:V>N	*talker*

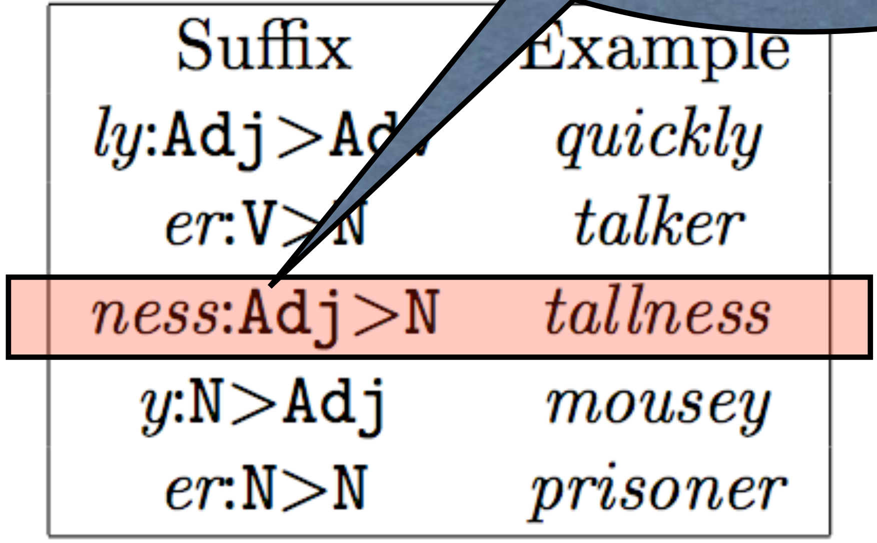
isting (MAG)



	Example
	quickly
	regression
	talker
dv	*bitingly*
y:N>Adj	*mousey*

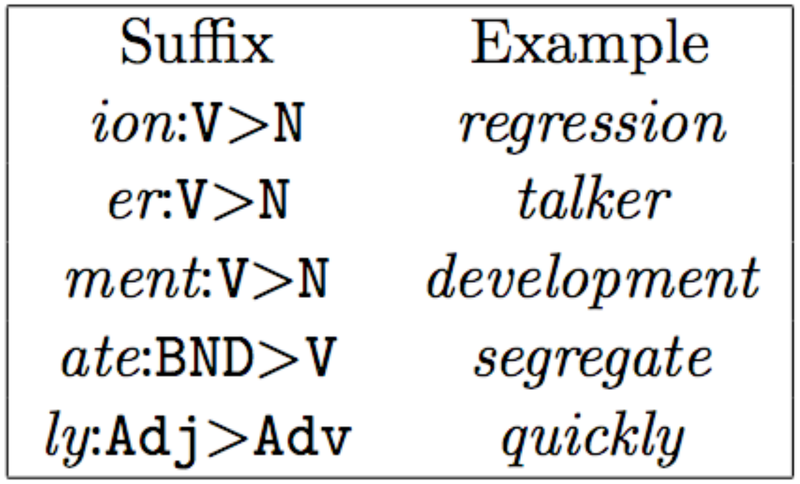
High Proportion of Low Frequency Types

Inference



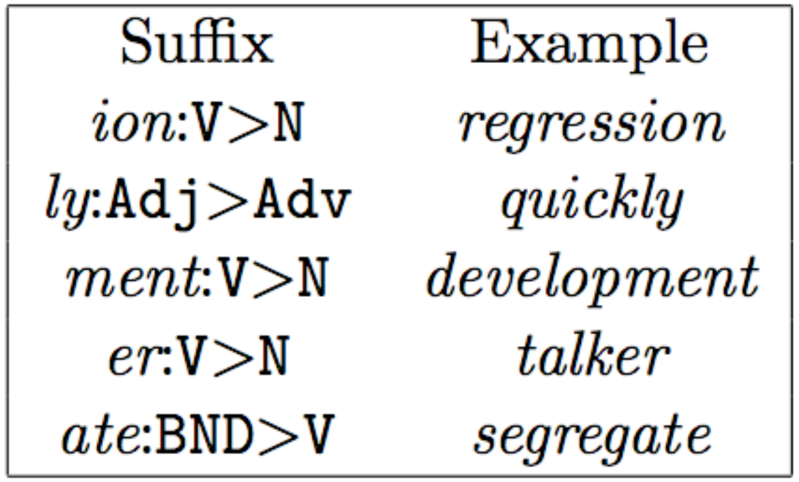
Suffix	Example
ly:Adj>Adv	*quickly*
er:V>N	*talker*
ness:Adj>N	*tallness*
y:N>Adj	*mousey*
er:N>N	*prisoner*

Exemplar (DOP1)



Suffix	Example
ion:V>N	*regression*
er:V>N	*talker*
ment:V>N	*development*
ate:BND>V	*segregate*
ly:Adj>Adv	*quickly*

Exemplar (GDMN)



Suffix	Example
ion:V>N	*regression*
ly:Adj>Adv	*quickly*
ment:V>N	*development*
er:V>N	*talker*
ate:BND>V	*segregate*

Top 5 Most Productive Suffixes

Full-Parsing (M

High Token Frequency

Suffix	
ion:V>N	
ly:Adj>Adv	
ate:BND>V	*segregate*
ment:V>N	*development*
er:V>N	*talker*

Exemp

High Token Frequency

Suffix	Exa
ion:V>N	*regression*
er:V>N	*talker*
ment:V>N	*development*
ate:BND>V	*segregate*
ly:Adj>Adv	*quickly*

Inference-Based (FG)

Suffix	Example
Adj>Adv	*quickly*
V>N	*talker*
Adj>N	*tallness*
y:N>Adj	*mousey*
er:N>N	*prisoner*

Fu

High Type Frequency

Suffix	Example
ly:Adj>Adv	*quickly*
ion:V>N	*regression*
er:V>N	*talker*
ly:V>A	
y:N>	

High Token Frequency

Exemplar (GDMN)

Suffix	Example
ion:V>N	*regression*
ly:Adj>Adv	*quickly*
ment:V>N	*development*
er:V>N	*talker*
ate:BND>V	*segregate*

Baayen's *Hapax*-Based Measures

- Baayen's $\mathcal{P}$ / $\mathcal{P}^*$ (e.g., Baayen, 1992)
  - Estimators of productivity based on the proportion of frequency-1 words in an input corpus.
  - Various derivations.
    - Rate of vocabulary change in urn model.
    - Good-Turing estimation.
    - Fundamentally, a rule-of-thumb.
  - Only defined for single affix estimation.

Productivity Correlations

($\mathcal{P}/\mathcal{P}^*$values from Hay & Baayen, 2002)

Measure	FG *(Inference)*	MDPCFG *(Full-parsing)*	MAG *(Full-listing)*	DOP1 *(Exemplar-based)*	ENDOP *(Exemplar-based)*
$\mathcal{P}$	**0.907**	-0.0003	0.692	0.346	0.143
$\mathcal{P}^*$	**0.662**	0.480	0.568	0.402	0.500

Fragment Grammars and Hapaxes

- For the case of single affixes, Fragments Grammars behave approximately as if they were using hapaxes.
- Not an explicit assumption of the model
- Model is about how words are built. Given the fact that some new words are built, behavior arises automatically.
- Generalizes to multi-way competition.

Case Studies

- Other approaches to productivity and reuse.

1. What distributions signal productivity?
2. How is competition resolved?
3. Multi-way competition.

Empirical Domains

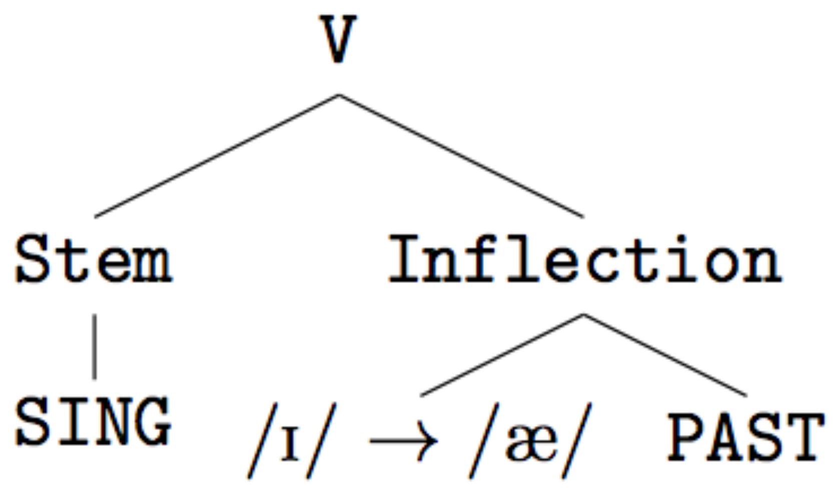
	Past Tense	Derivational Morphology
Productive	+ed *(walked)*	+ness *(goodness)*
Context-Dependent	ɪ → æ *(sang)*	+ity *(ability)*
Unproductive	suppletion *(go/went)*	+th *(width)*

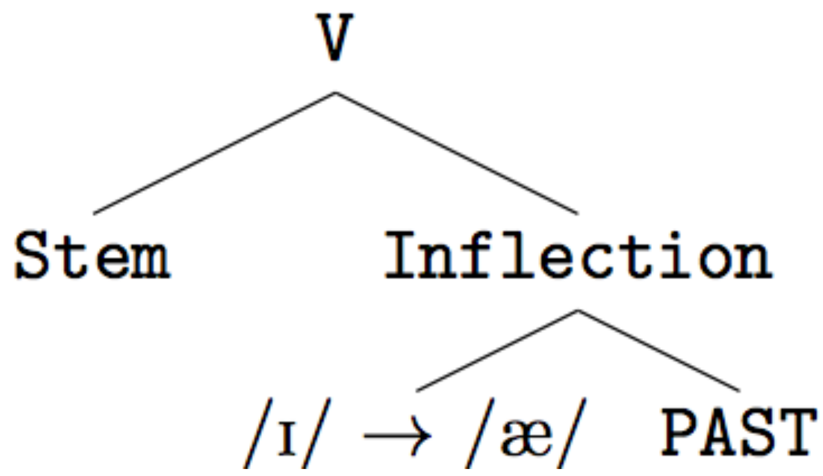
Crucial Facts

- **Defaultness**: Regular rule applies when all else fails.
- **Blocking**: Existence of irregular blocks regular rule.
- In this domain preferences are sharp.

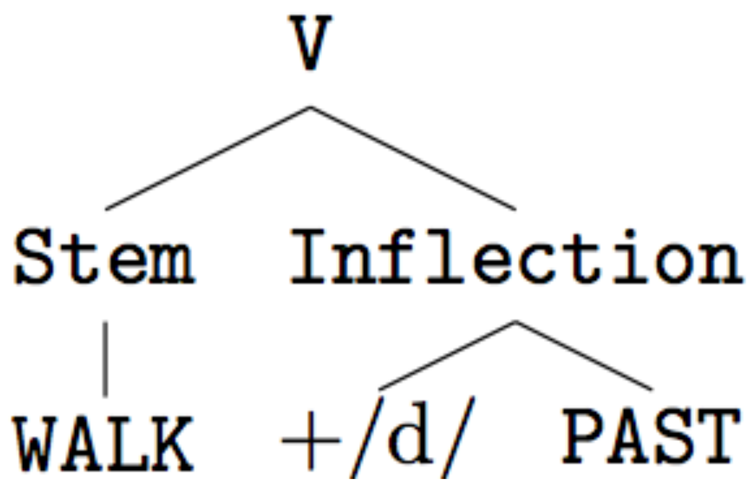
How can Correct Inflection be Represented?

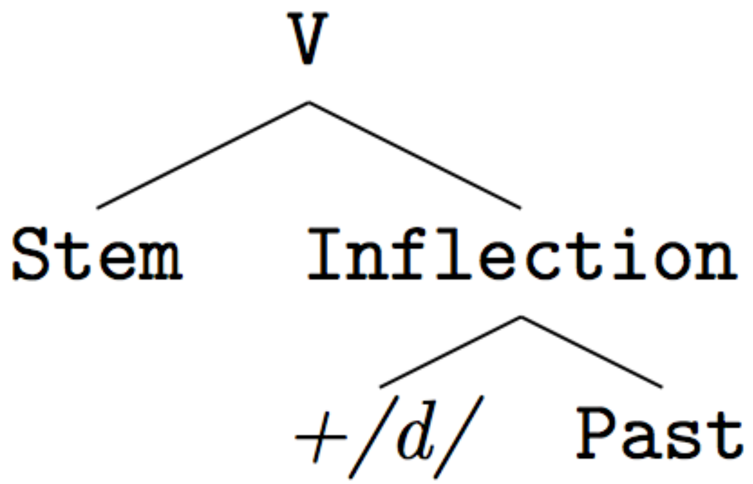
Irregulars





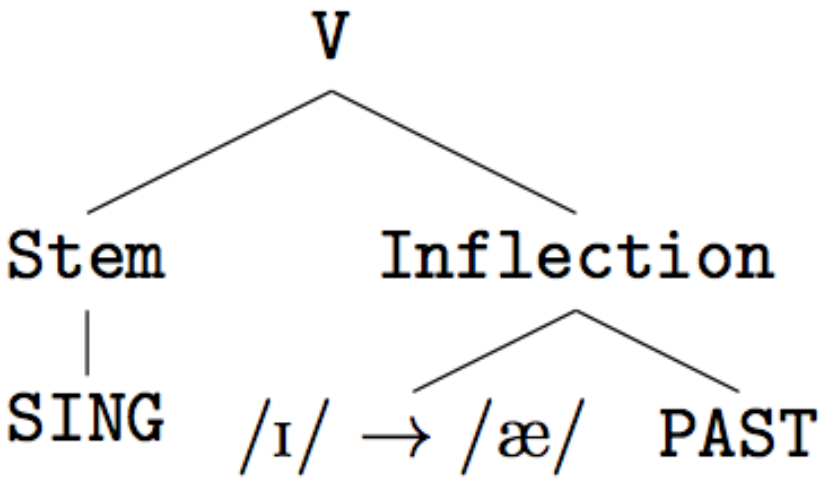
Regulars

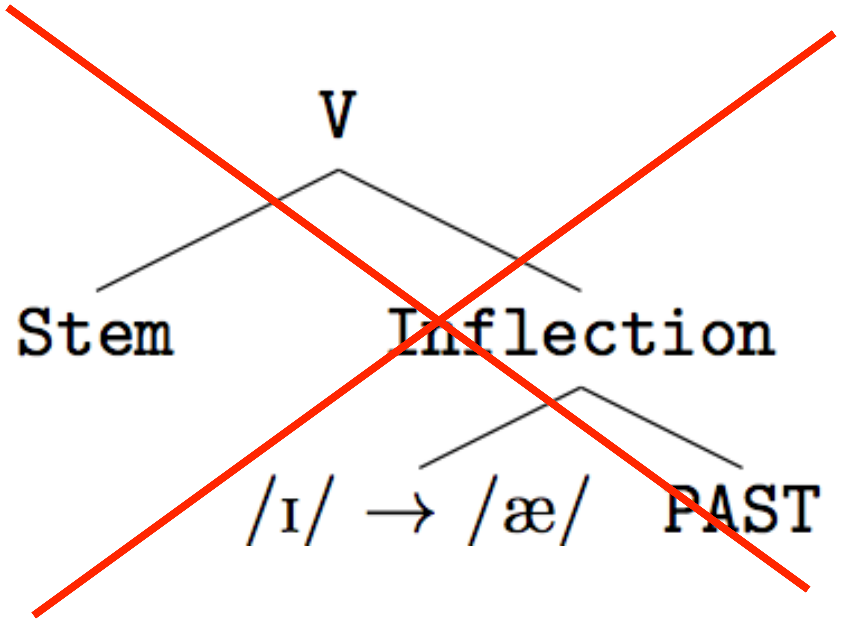




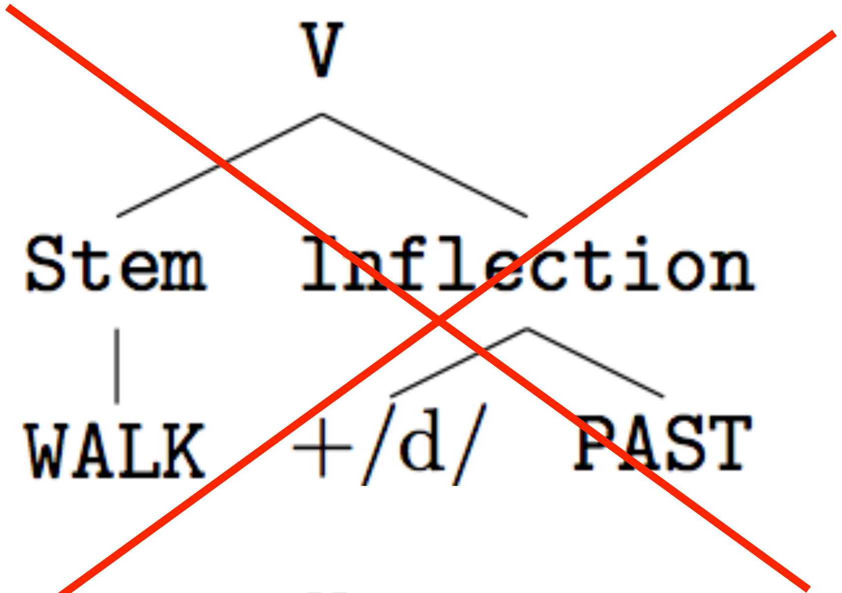
How can Correct Inflection be Represented?

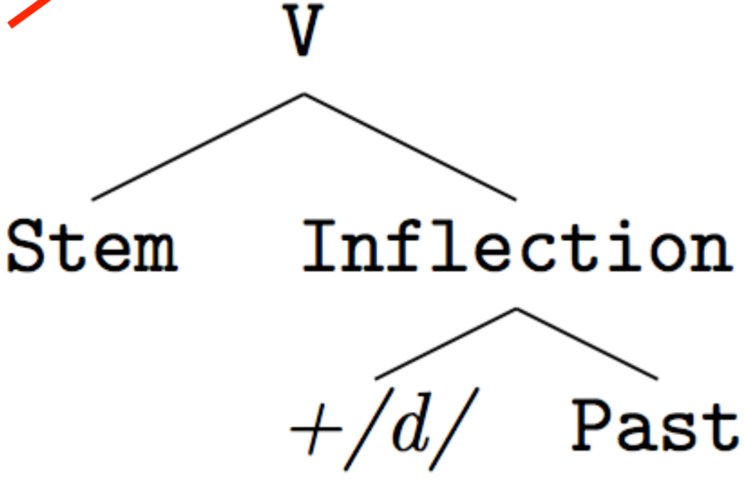
Irregulars



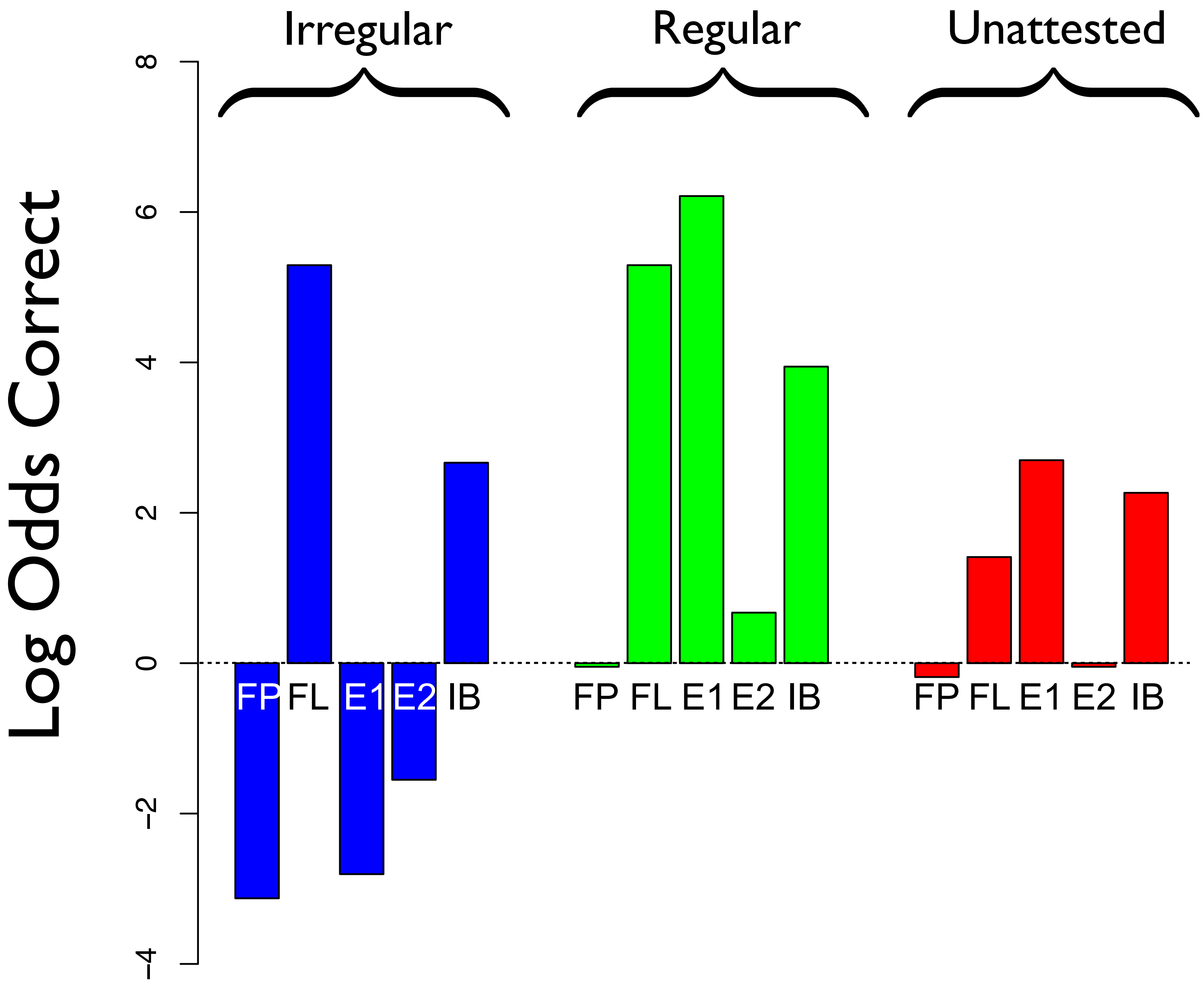


Regulars





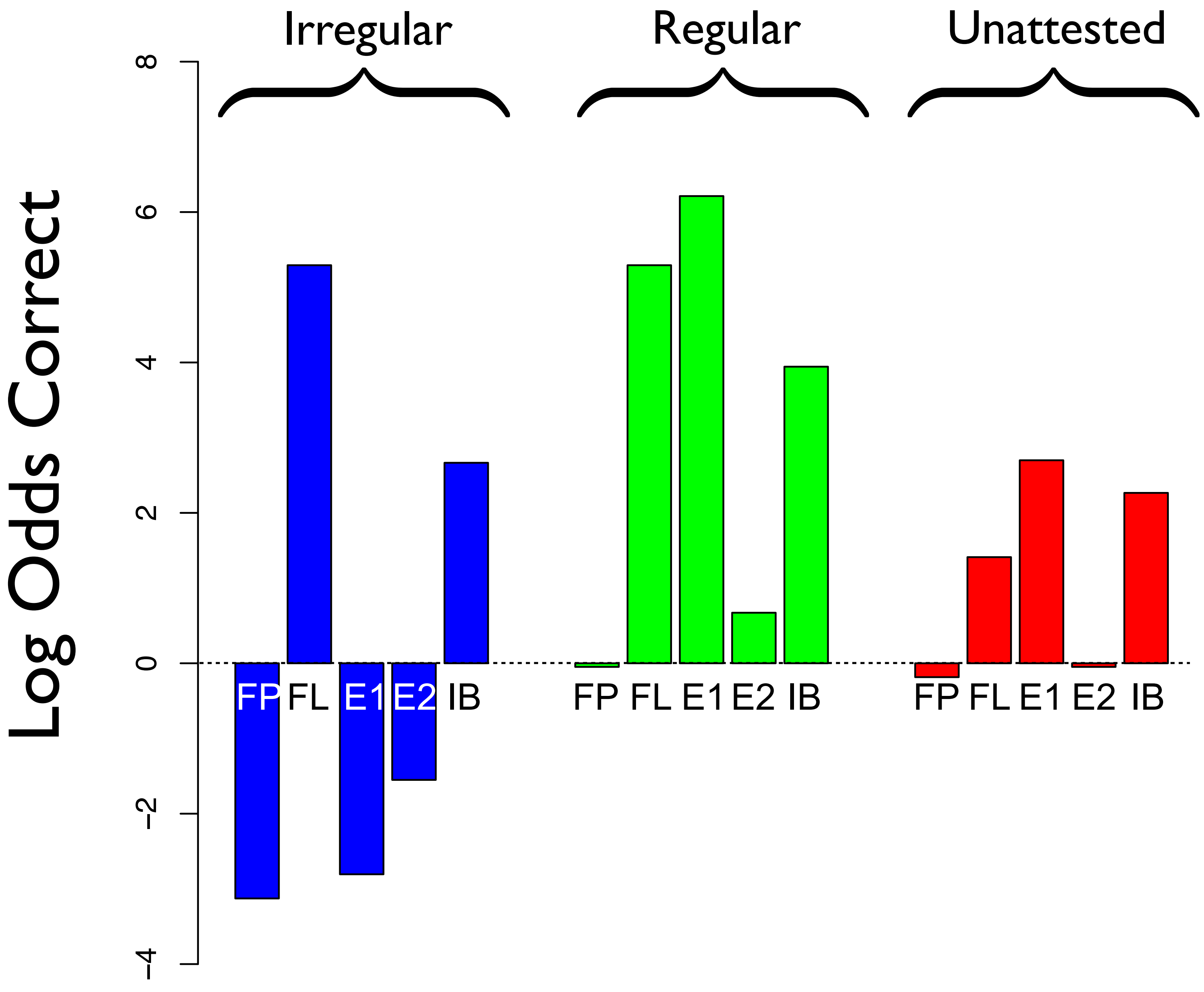
Correct Inflection



FP	*Full-Parsing* *(Multinomial-Dirichlet CFG)*
FL	*Full-Listing* *(Adaptor Grammars)*
E1	*Exemplar* *(Data-Oriented Parsing 1)*
E2	*Exemplar* *(DOP: ENDOP)*
IB	*Inference-Based* *(Fragment Grammars)*

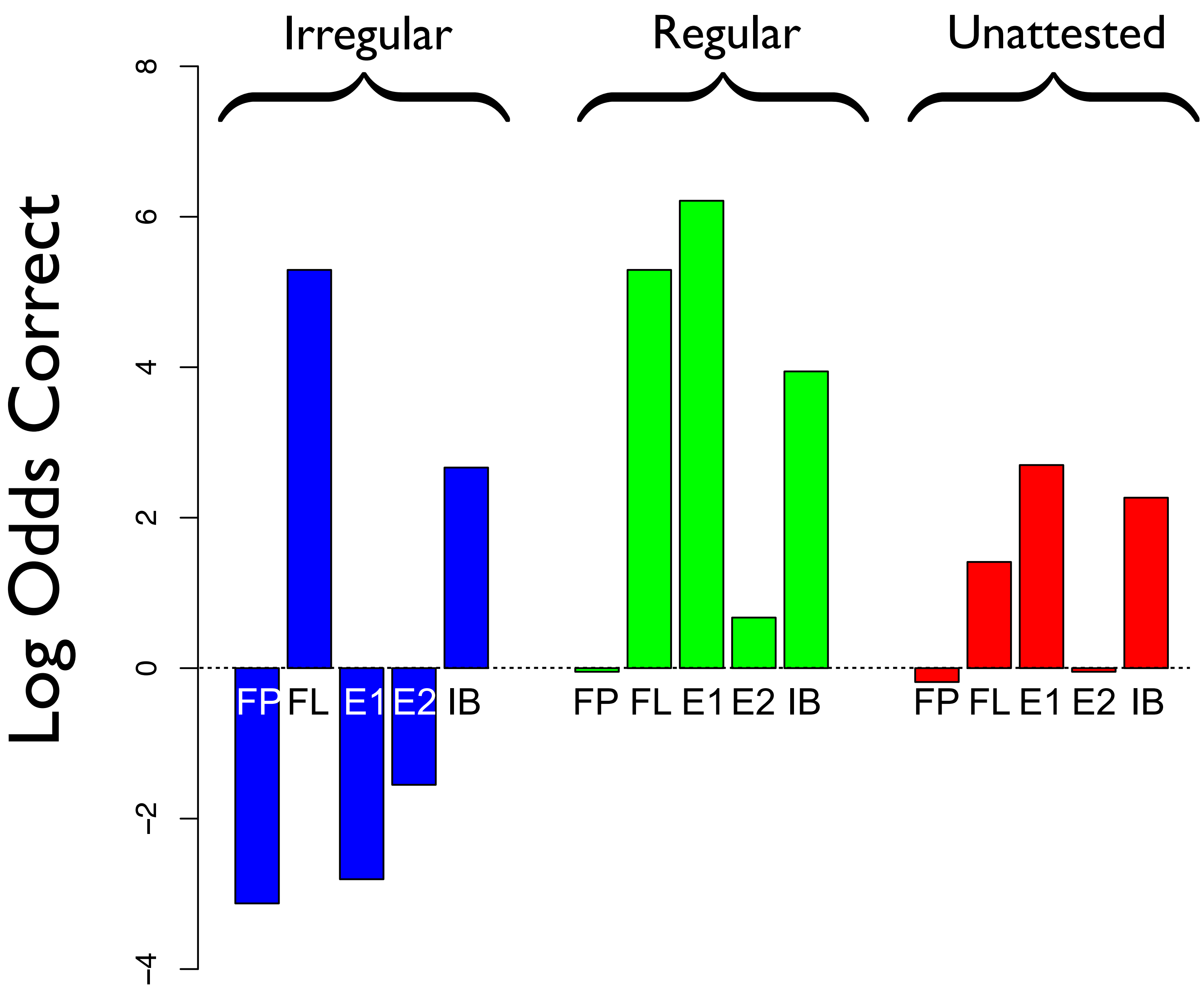
Correct Inflection

Preference
for
Correct
Past Form



FP	*Full-Parsing* *(Multinomial-Dirichlet CFG)*
FL	*Full-Listing* *(Adaptor Grammars)*
E1	*Exemplar* *(Data-Oriented Parsing 1)*
E2	*Exemplar* *(DOP: ENDOP)*
IB	*Inference-Based* *(Fragment Grammars)*

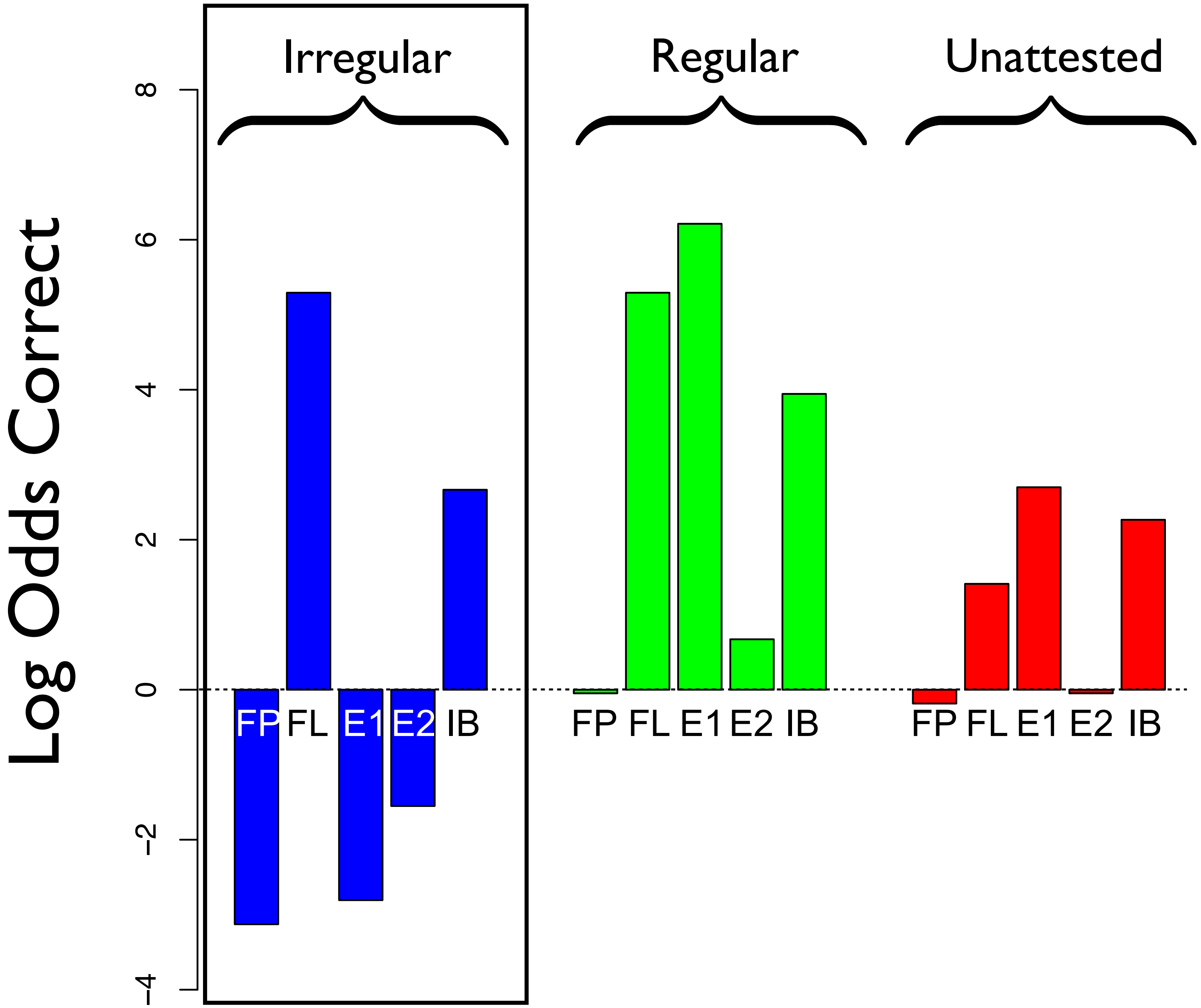
Correct Inflection



FP	*Full-Parsing* *(Multinomial-Dirichlet CFG)*
FL	*Full-Listing* *(Adaptor Grammars)*
E1	*Exemplar* *(Data-Oriented Parsing 1)*
E2	*Exemplar* *(DOP: ENDOP)*
IB	*Inference-Based* *(Fragment Grammars)*

Preference
for
Incorrect
Past Form

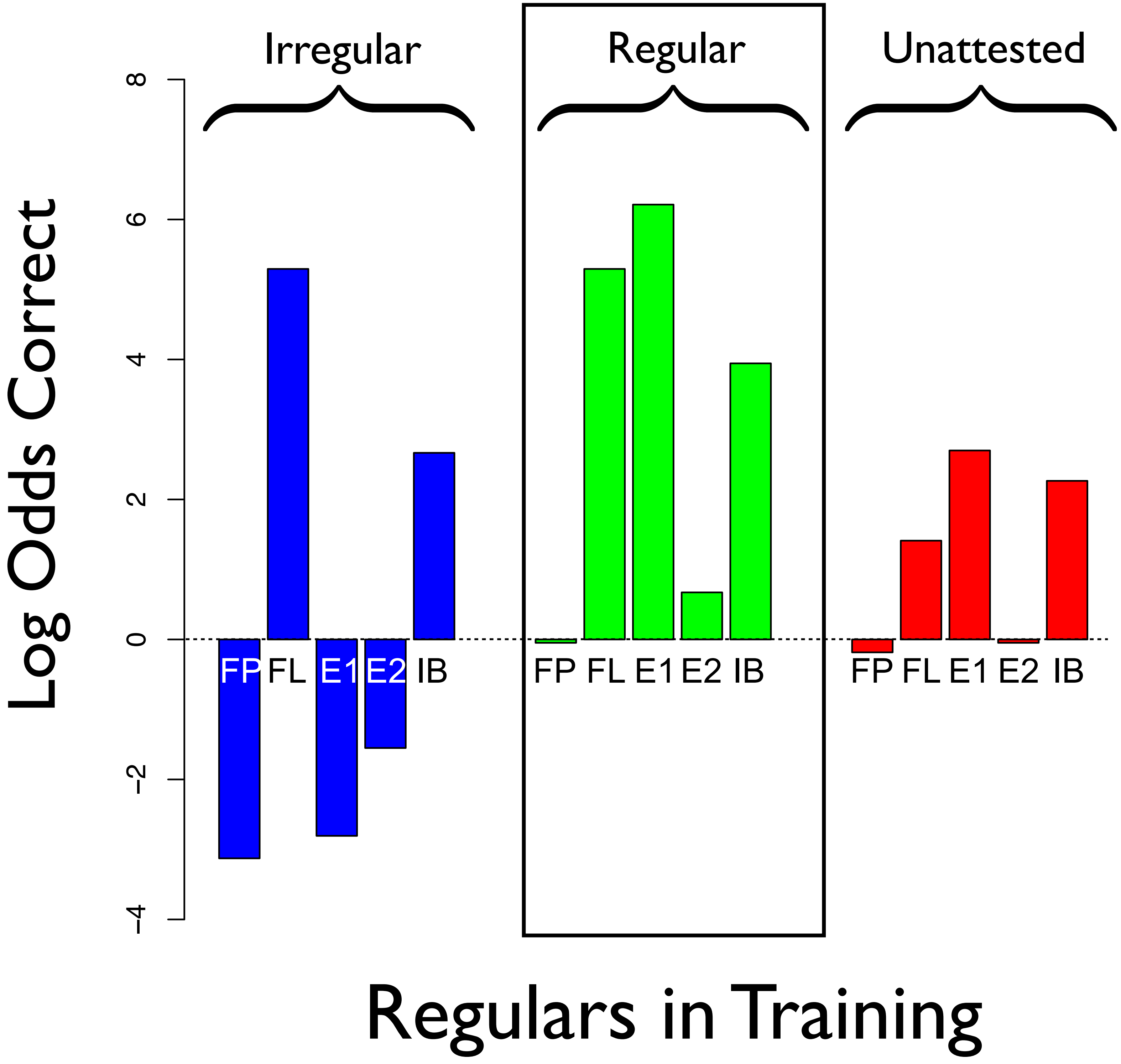
Correct Inflection



FP	*Full-Parsing* *(Multinomial-Dirichlet CFG)*
FL	*Full-Listing* *(Adaptor Grammars)*
E1	*Exemplar* *(Data-Oriented Parsing 1)*
E2	*Exemplar* *(DOP: ENDOP)*
IB	*Inference-Based* *(Fragment Grammars)*

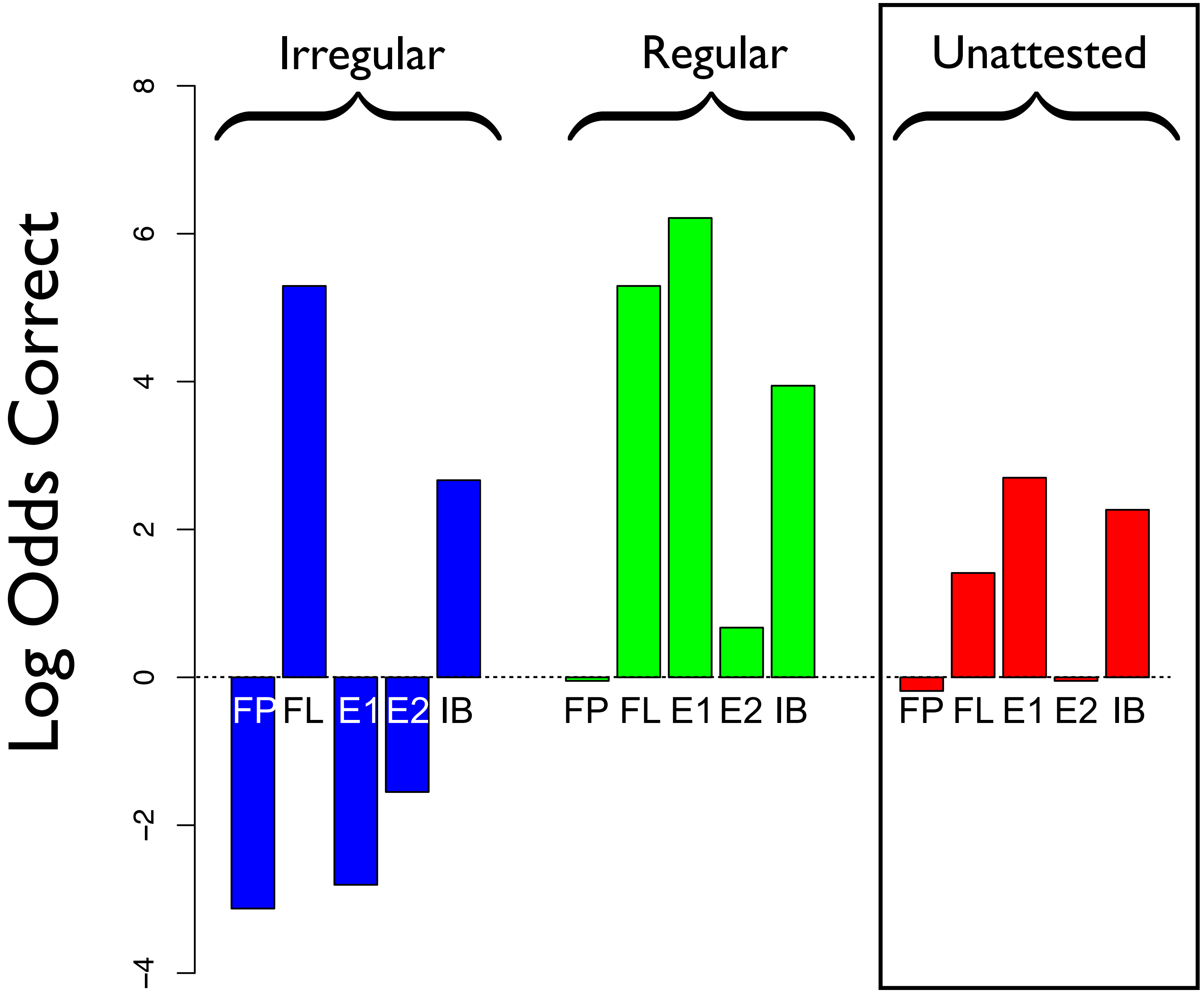
Irregulars in Training

Correct Inflection



FP	*Full-Parsing* *(Multinomial-Dirichlet CFG)*
FL	*Full-Listing* *(Adaptor Grammars)*
E1	*Exemplar* *(Data-Oriented Parsing 1)*
E2	*Exemplar* *(DOP: ENDOP)*
IB	*Inference-Based* *(Fragment Grammars)*

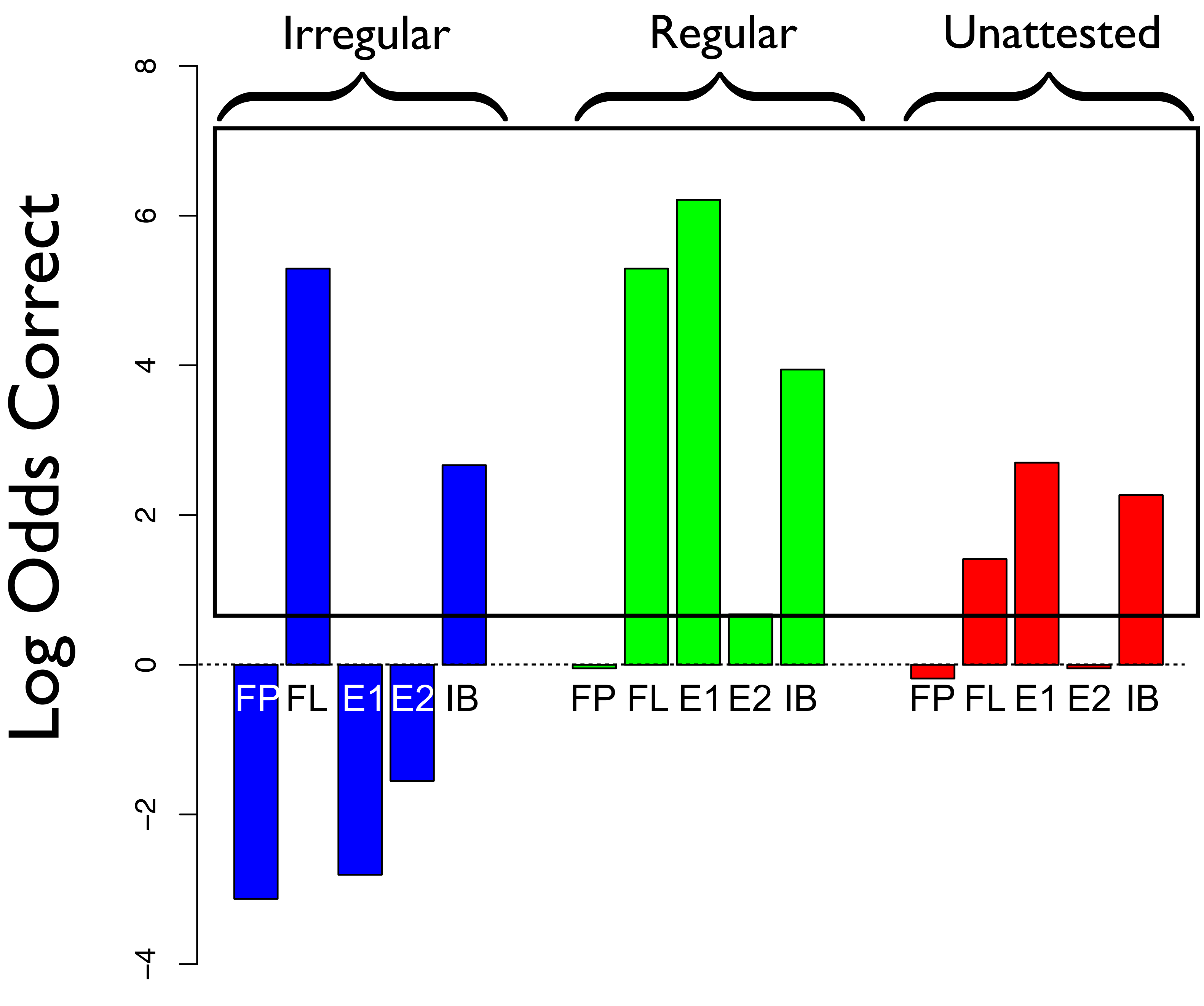
Correct Inflection



FP	*Full-Parsing* *(Multinomial-Dirichlet CFG)*
FL	*Full-Listing* *(Adaptor Grammars)*
E1	*Exemplar* *(Data-Oriented Parsing 1)*
E2	*Exemplar* *(DOP: ENDOP)*
IB	*Inference-Based* *(Fragment Grammars)*

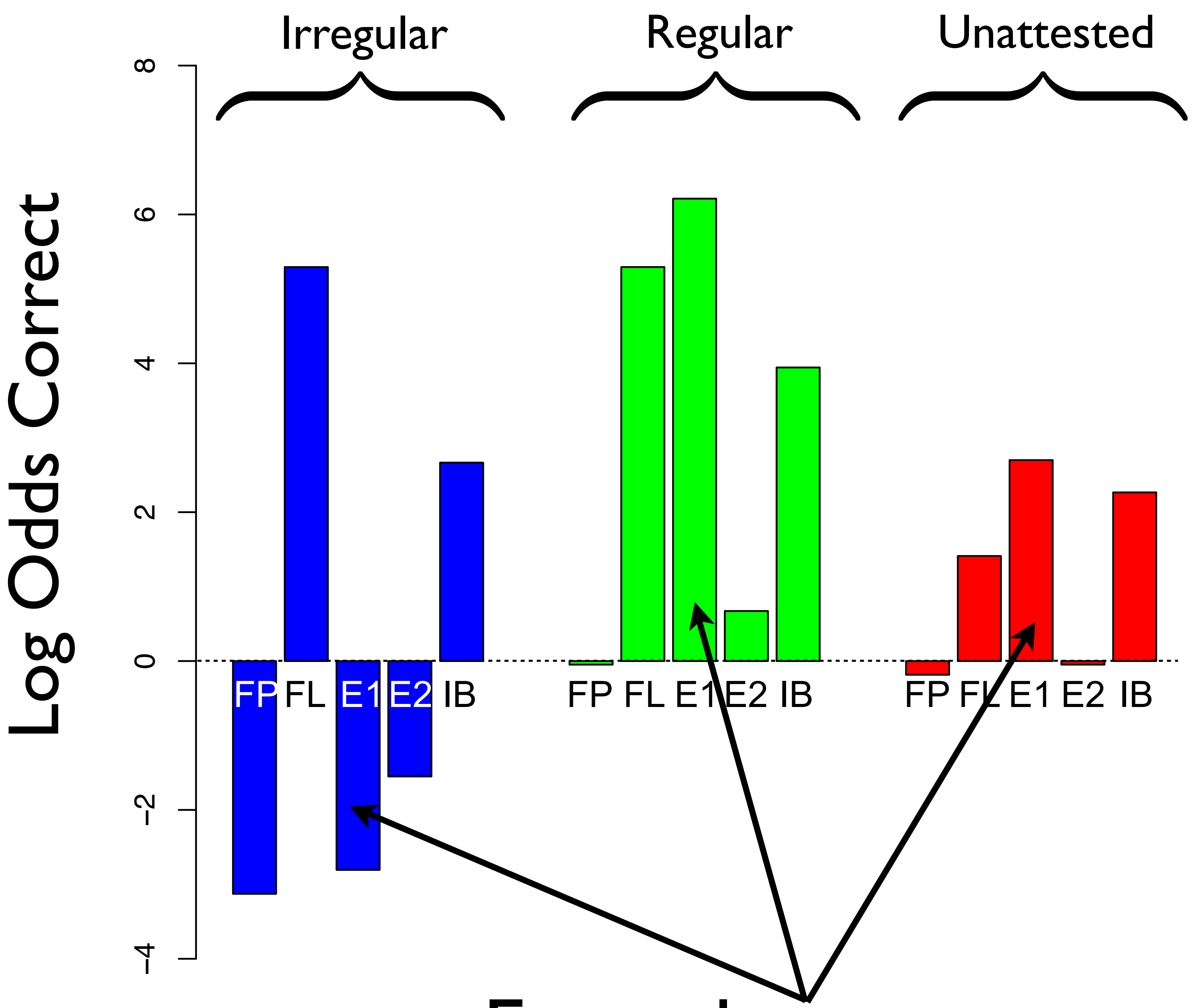
Regulars and Irregulars
not in Training

Correct Inflection



FP	*Full-Parsing* *(Multinomial-Dirichlet CFG)*
FL	*Full-Listing* *(Adaptor Grammars)*
E1	*Exemplar* *(Data-Oriented Parsing 1)*
E2	*Exemplar* *(DOP: ENDOP)*
IB	*Inference-Based* *(Fragment Grammars)*

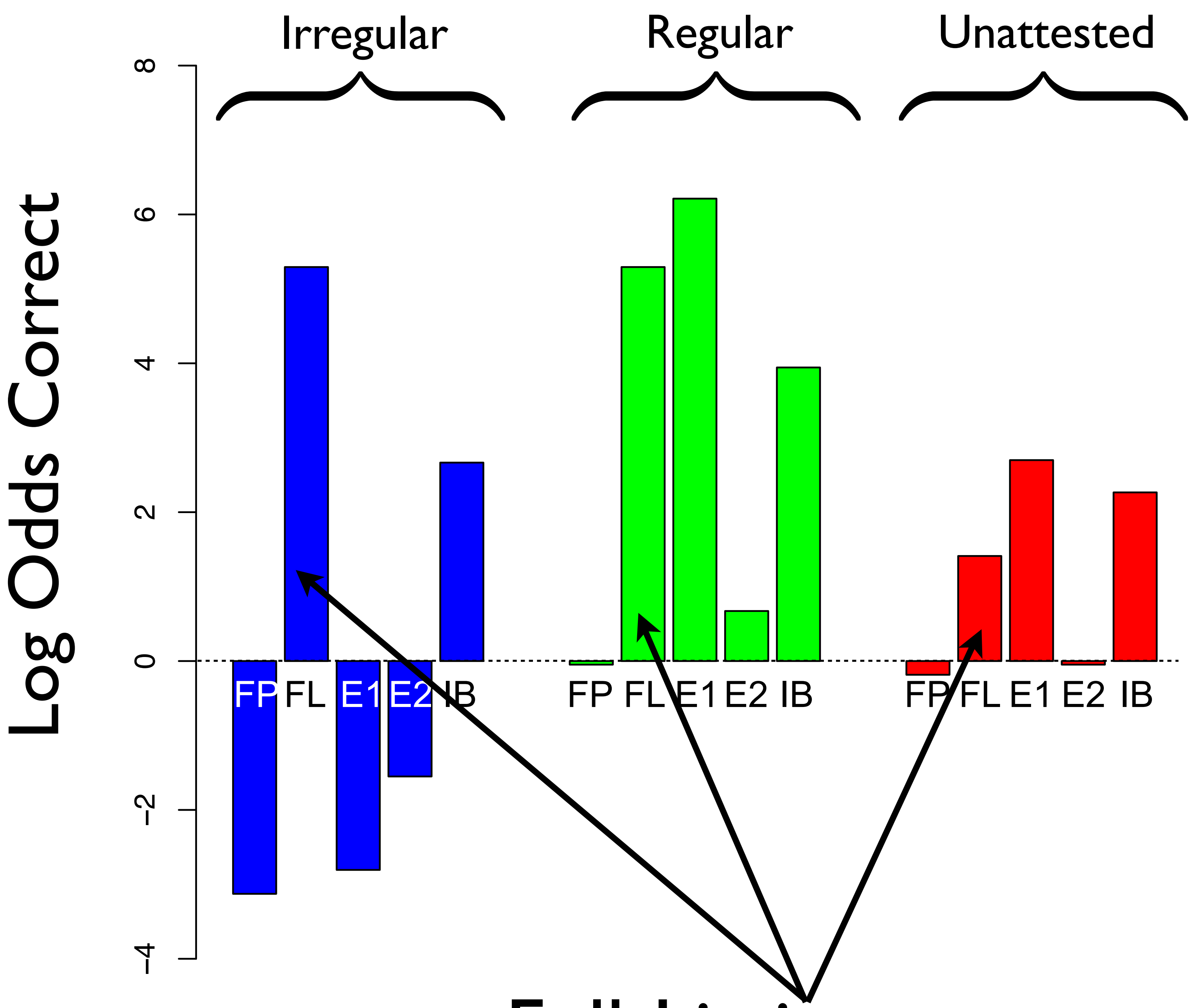
Correct Inflection



FP	*Full-Parsing* *(Multinomial-Dirichlet CFG)*
FL	*Full-Listing* *(Adaptor Grammars)*
E1	*Exemplar* *(Data-Oriented Parsing 1)*
E2	*Exemplar* *(DOP: ENDOP)*
IB	*Inference-Based* *(Fragment Grammars)*

Exemplar *(Data-Oriented Parsing 1)*

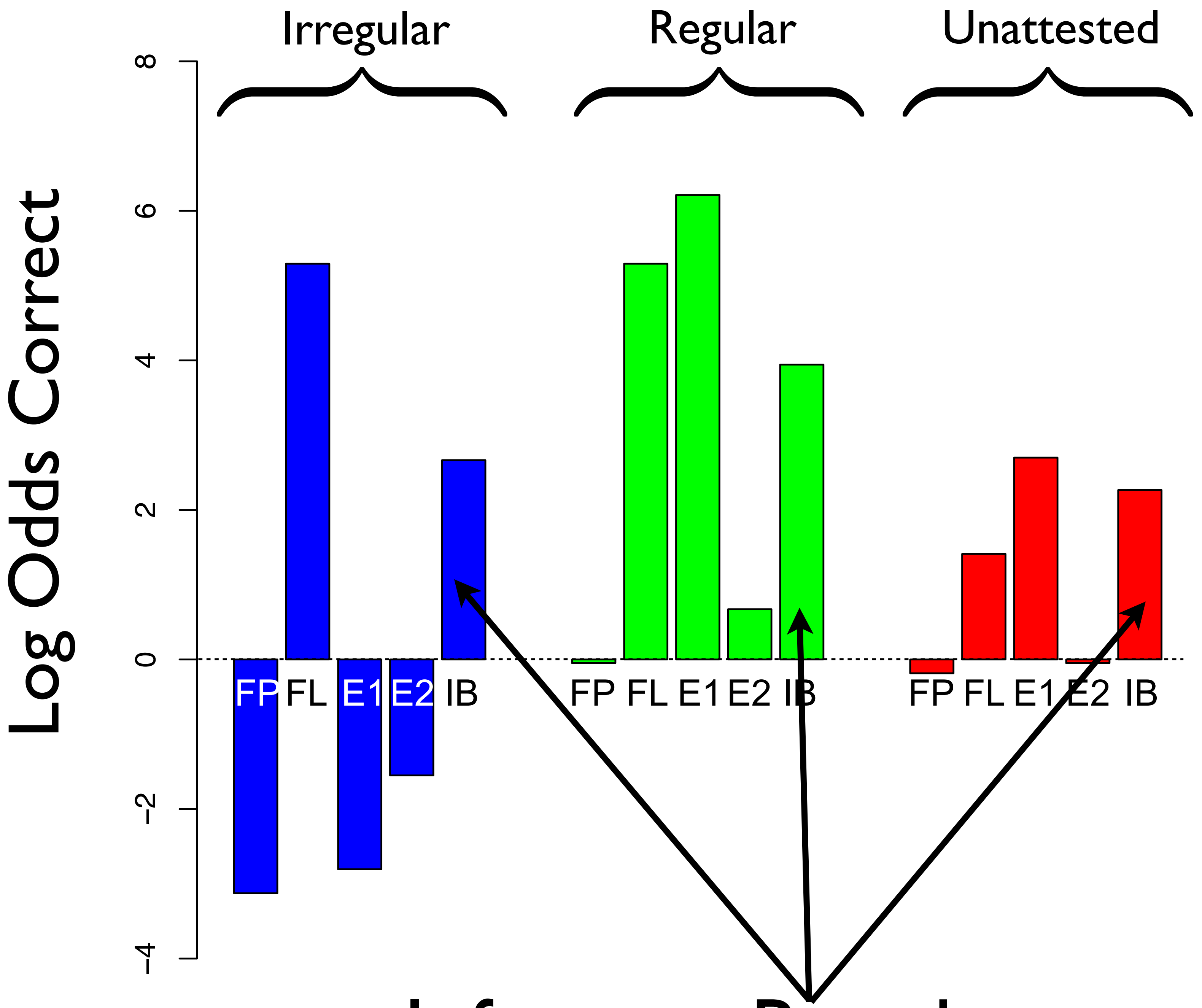
Correct Inflection



Full-Listing *(Adaptor Grammars)*

FP	*Full-Parsing* *(Multinomial-Dirichlet CFG)*
FL	*Full-Listing* *(Adaptor Grammars)*
E1	*Exemplar* *(Data-Oriented Parsing 1)*
E2	*Exemplar* *(DOP: ENDOP)*
IB	*Inference-Based* *(Fragment Grammars)*

Correct Inflection



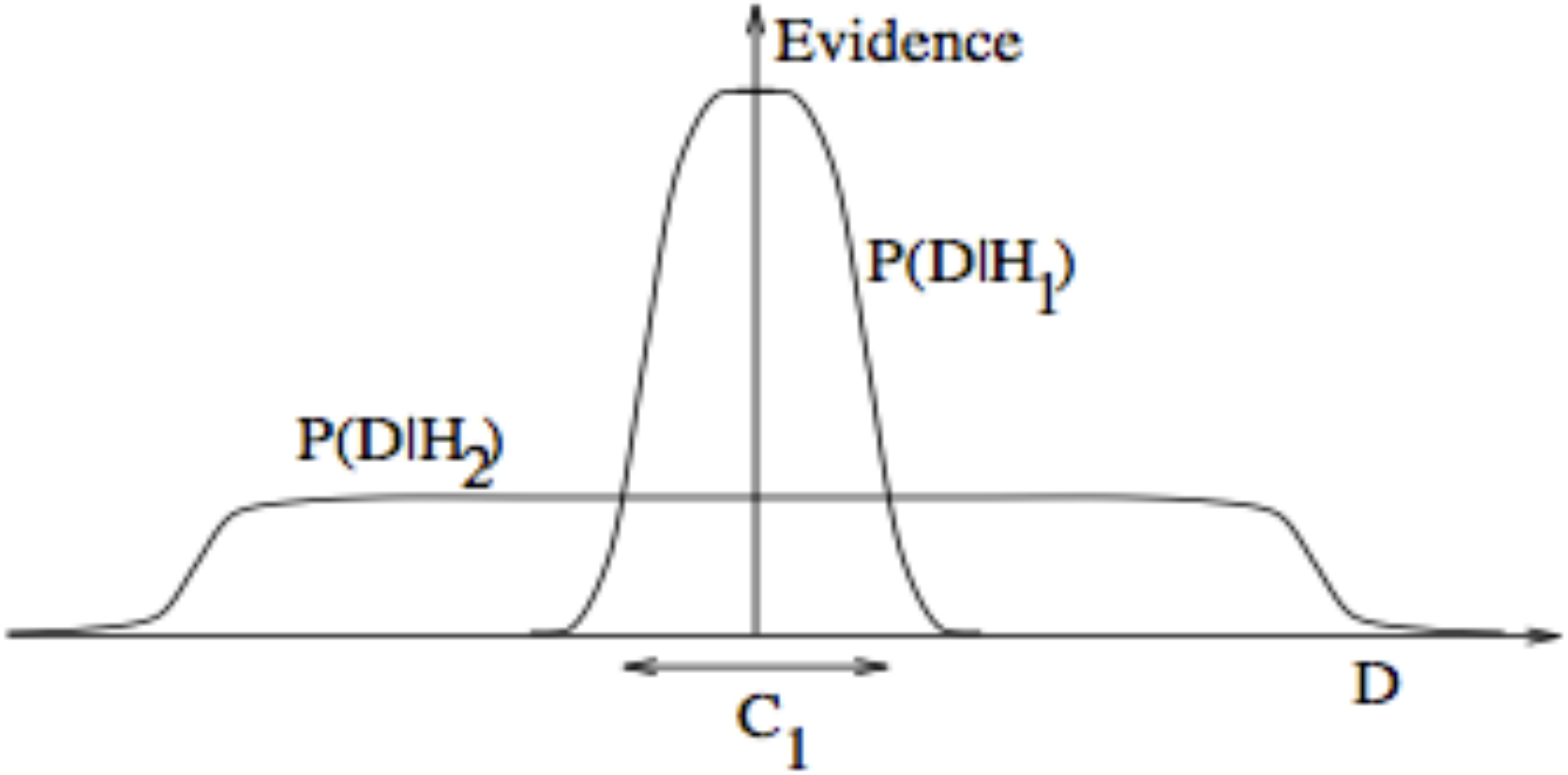
FP	*Full-Parsing* *(Multinomial-Dirichlet CFG)*
FL	*Full-Listing* *(Adaptor Grammars)*
E1	*Exemplar* *(Data-Oriented Parsing 1)*
E2	*Exemplar* *(DOP: ENDOP)*
IB	*Inference-Based* *(Fragment Grammars)*

Inference-Based *(Fragment Grammars)*

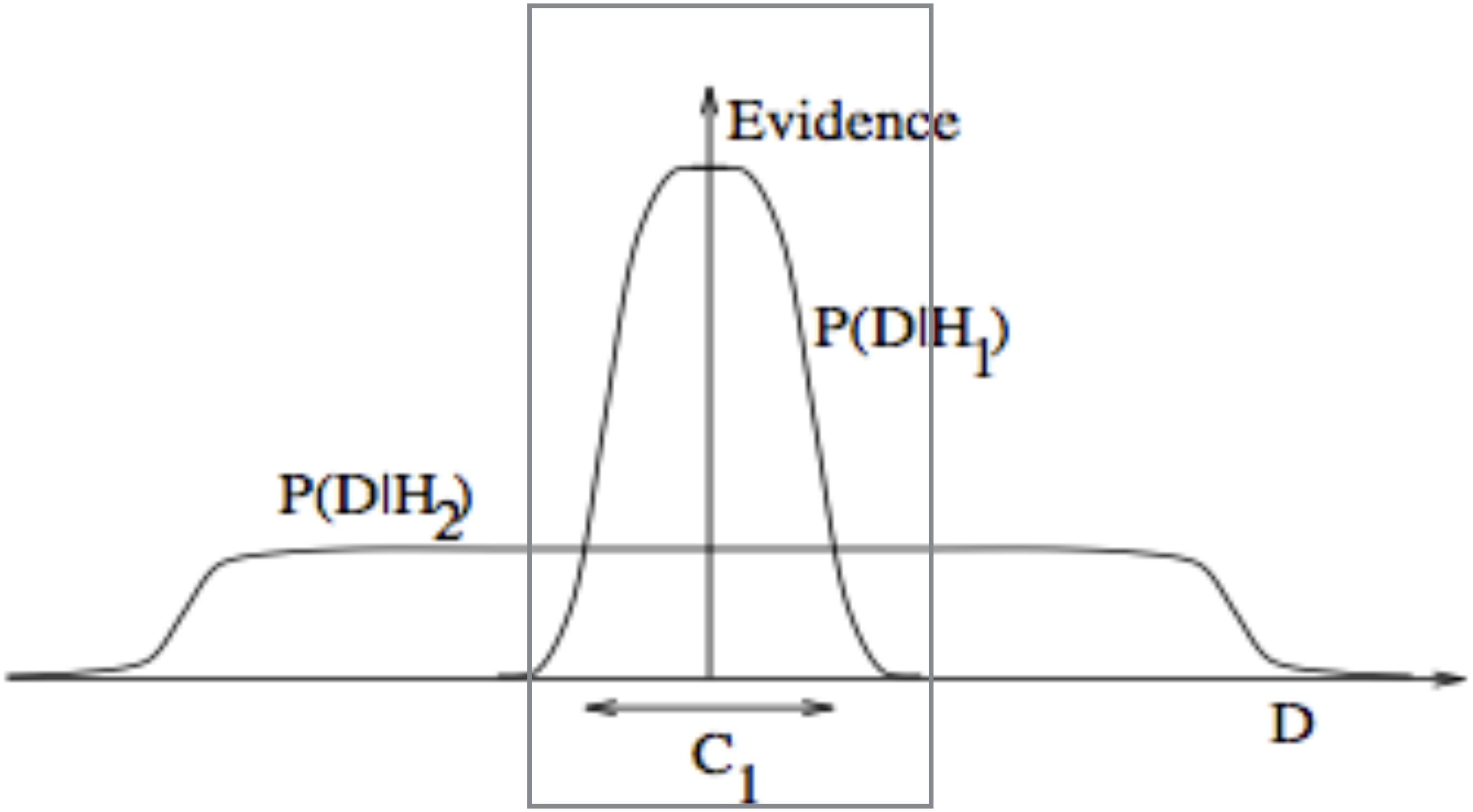
Why Does Blocking Occur?

- Consequence of two principles.
- **Law of Conservation of Belief**: Hypotheses that predict a greater variety of observed datasets place less probability on each.
- **Conservativity of Conditioning**: Posterior distributions have same relative probability as prior distributions.

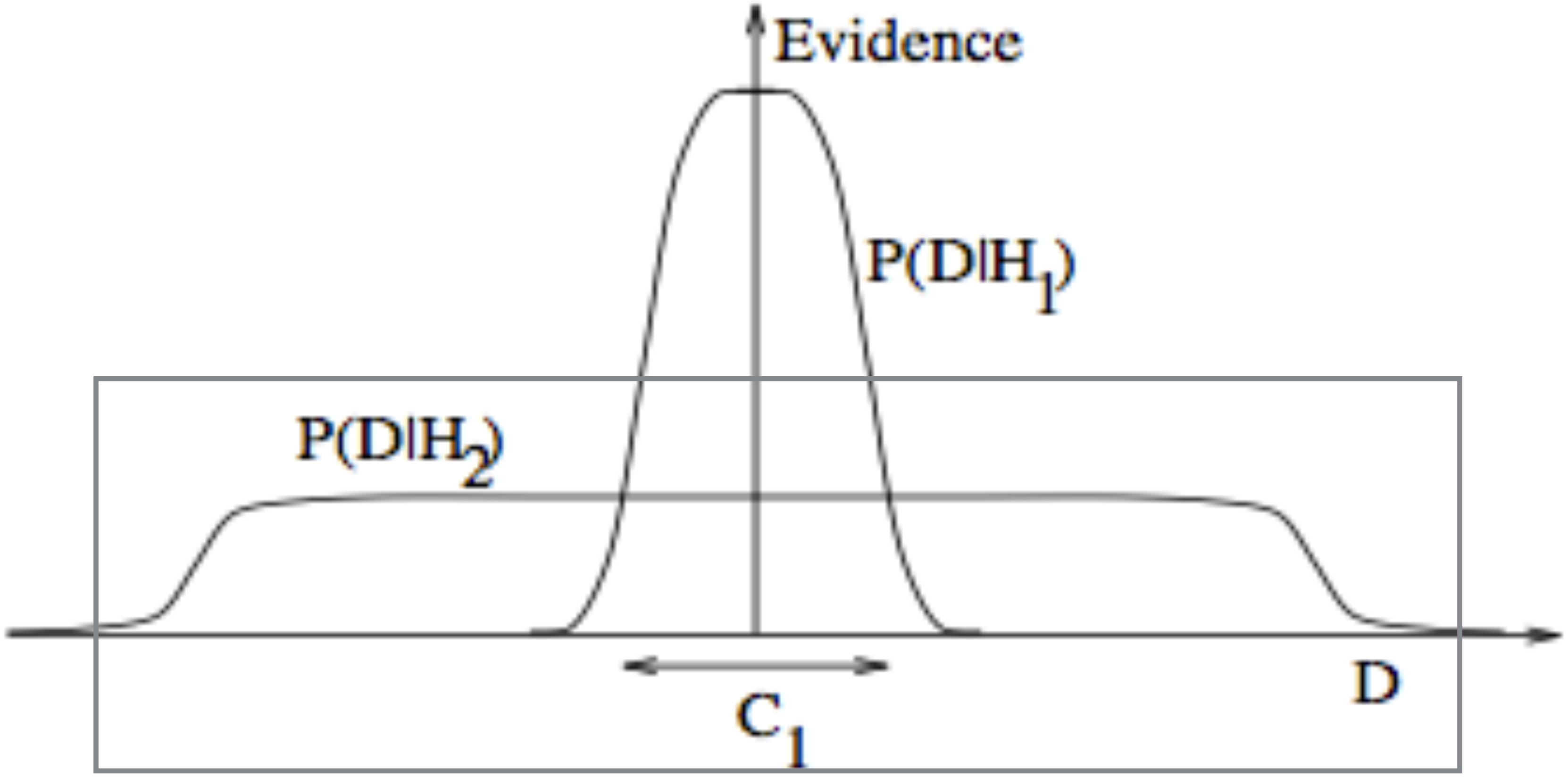
Law of Conservation of Belief



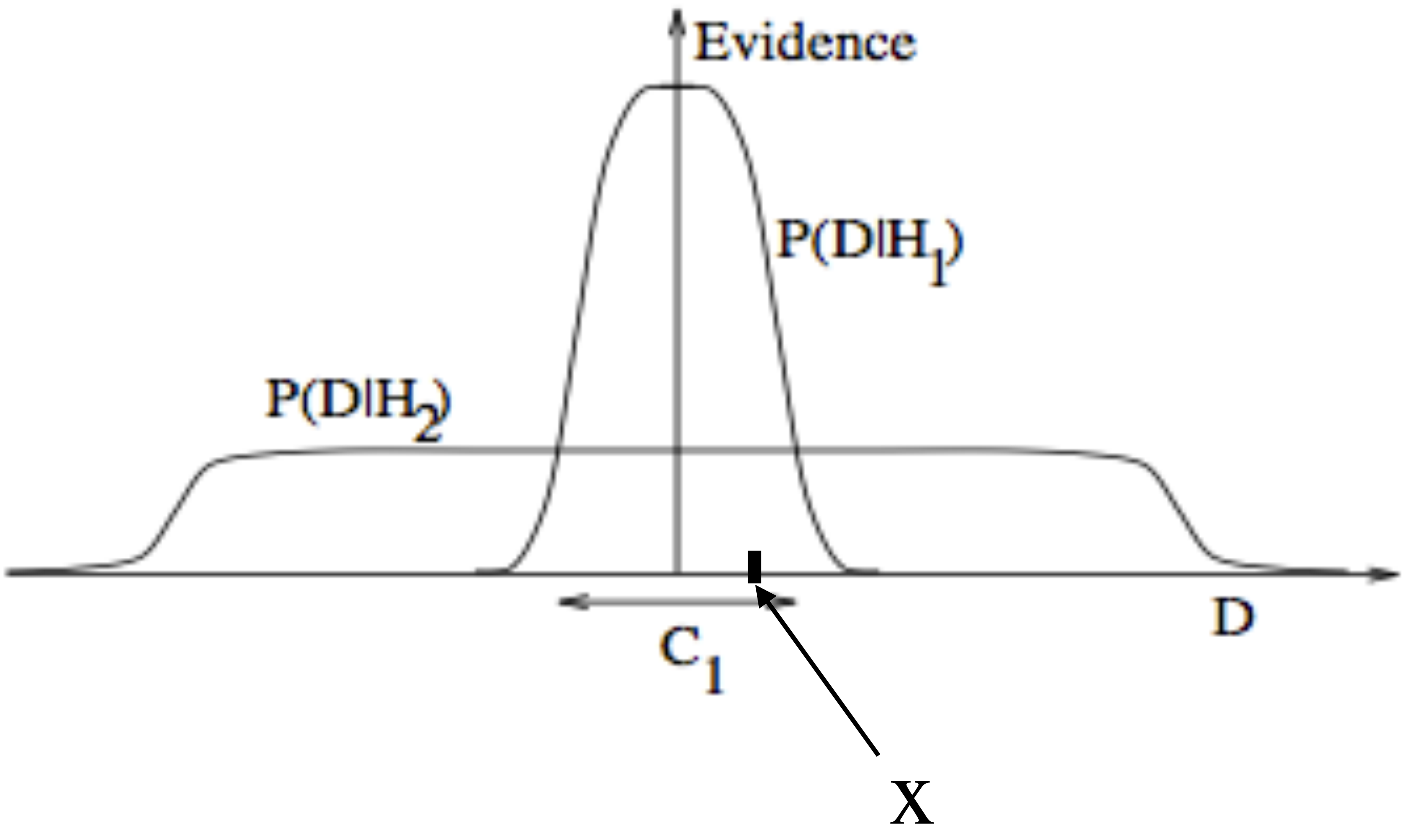
Law of Conservation of Belief



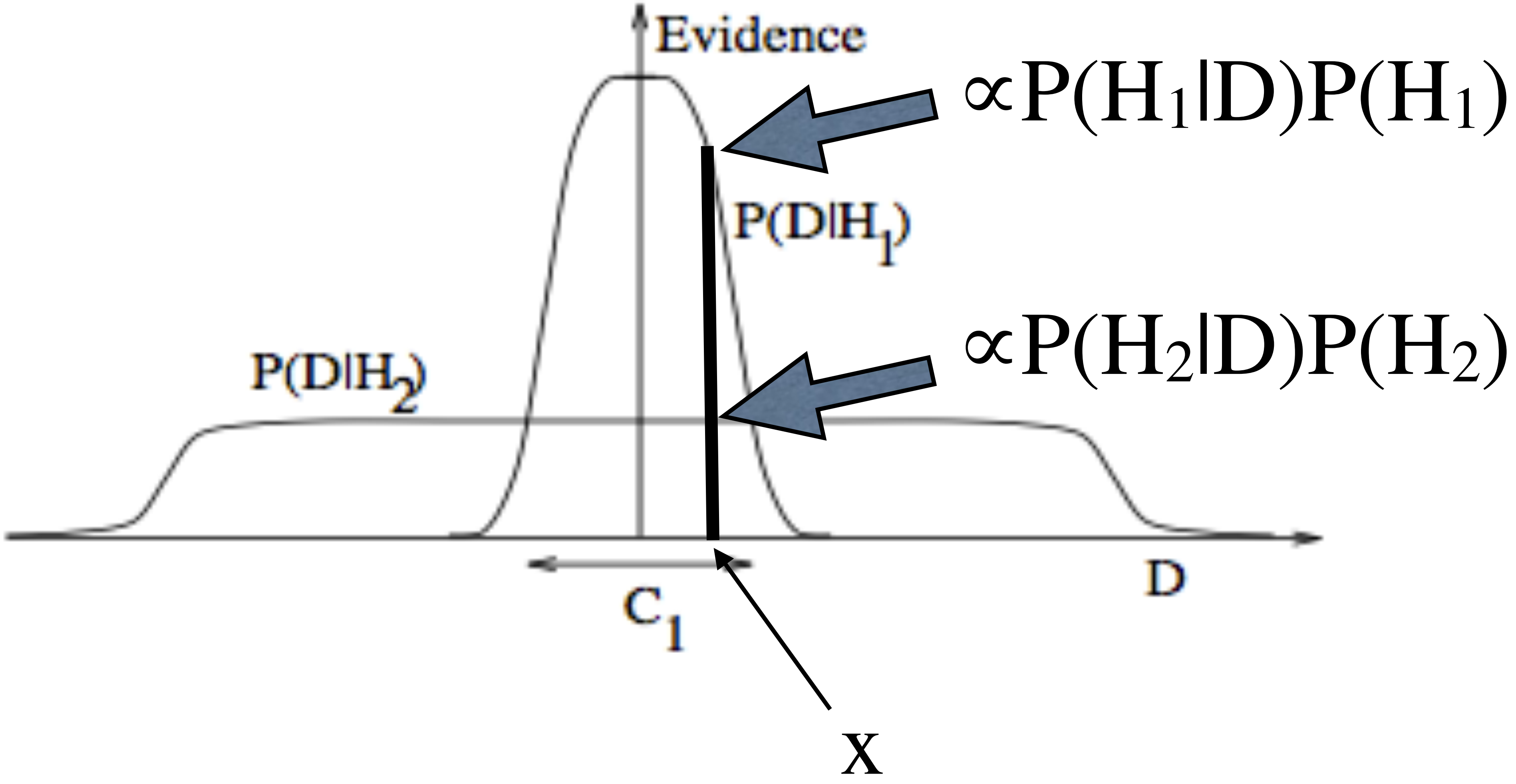
Law of Conservation of Belief



Observation



Conservativity

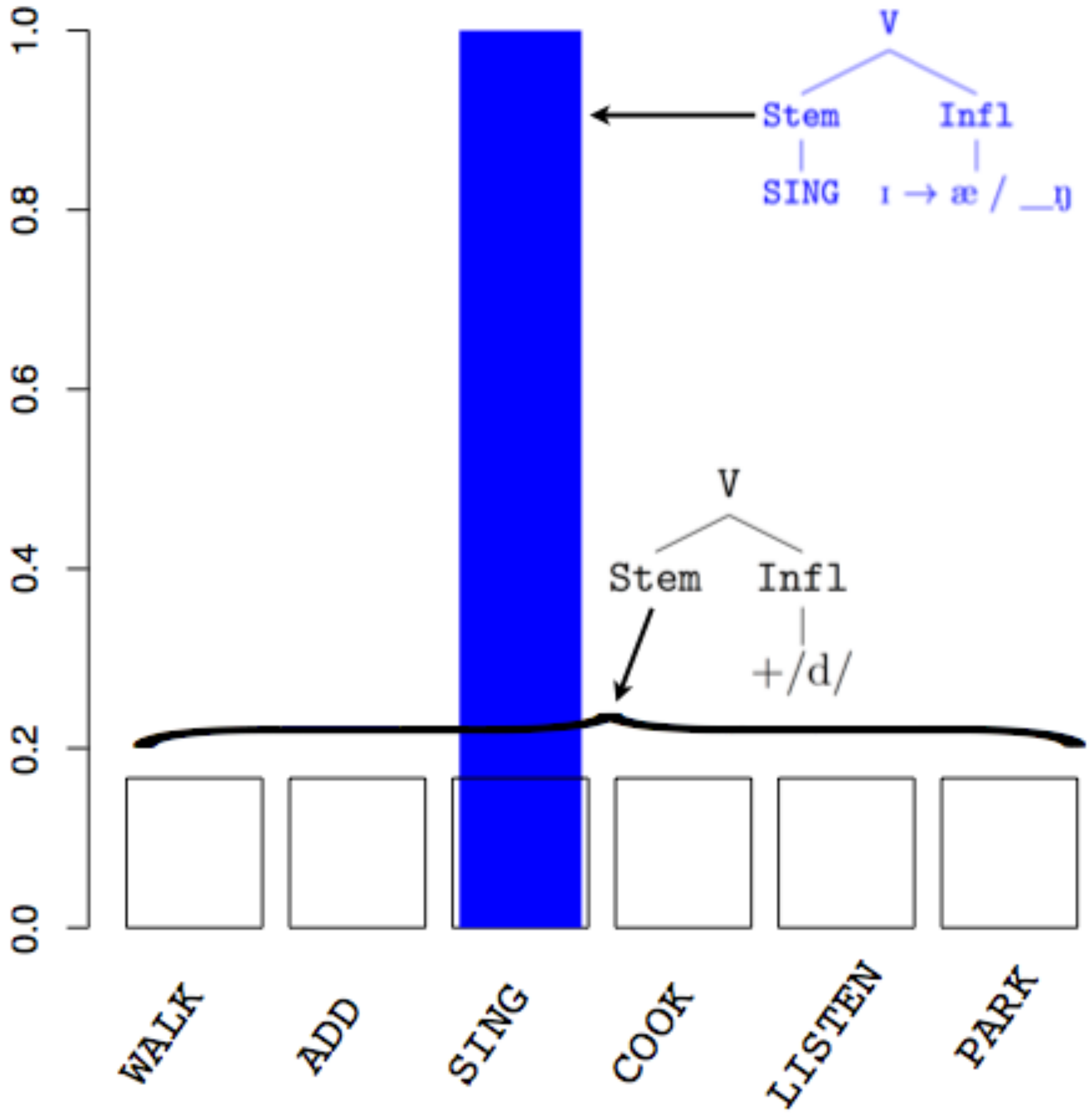


$\propto P(H_1|D)P(H_1)$

$\propto P(H_2|D)P(H_2)$

x

Past Tense


V
Stem
Infl
SING
ɪ → æ / __ŋ
V
Stem
Infl
+/d/
1.0
0.8
0.6
0.4
0.2
0.0
WALK
ADD
SING
COOK
LISTEN
PARK

Elsewhere

(Kiparsky, 1973; Anderson, 1969; Kiparsky, 1982a; Andrews, 1982)

- Don’t need *elsewhere condition* as independent stipulation (cf. *subset principle*, *premption*, etc.).
- When a choice must be made between two analyses/ derivations, prefer the one with highest P(form | meaning) more “tightly.”
- More general than original statement.
  - Any factor influencing P(form | meaning)
    - input conditions on rules, frequency, etc.
  - Stored-stored, stored-computed, computed-computed, etc.

Case Studies

- Other approaches to productivity and reuse.

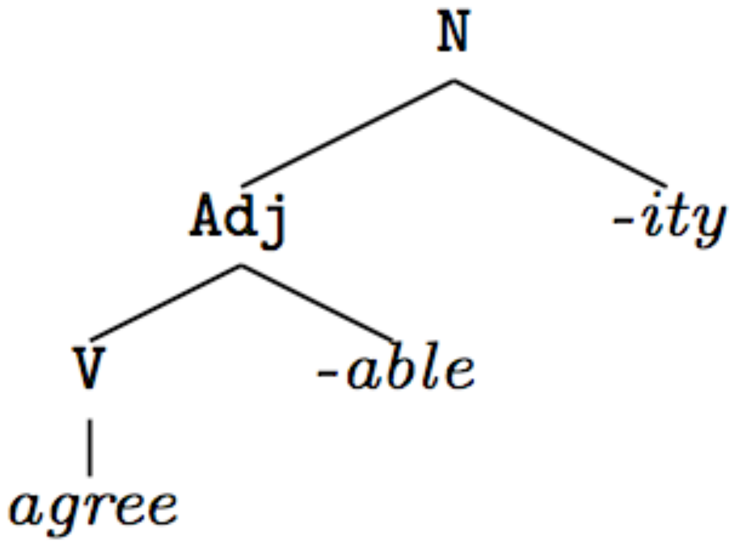
1. What distributions signal productivity?
2. How is competition resolved?
3. Multi-way competition.

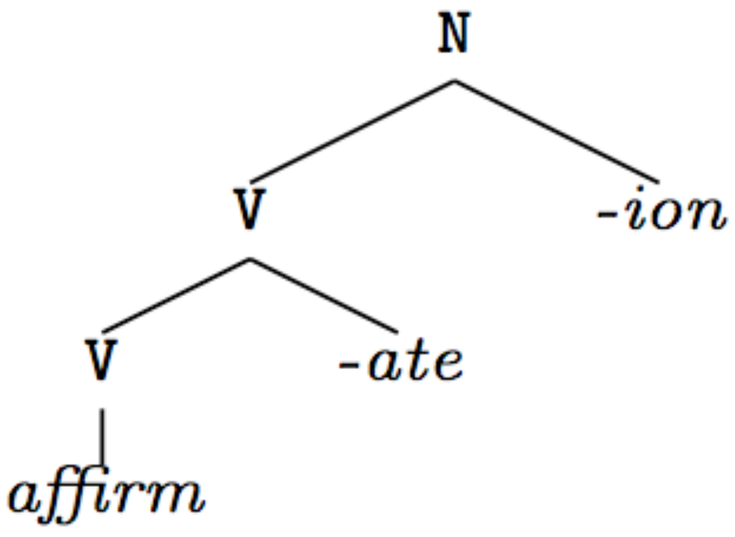
Empirical Domains

	Past Tense	Derivational Morphology
Productive	+ed *(walked)*	+ness *(goodness)*
Context-Dependent	ɪ →æ *(sang)*	+ity *(ability)*
Unproductive	suppletion *(go/went)*	+th *(width)*

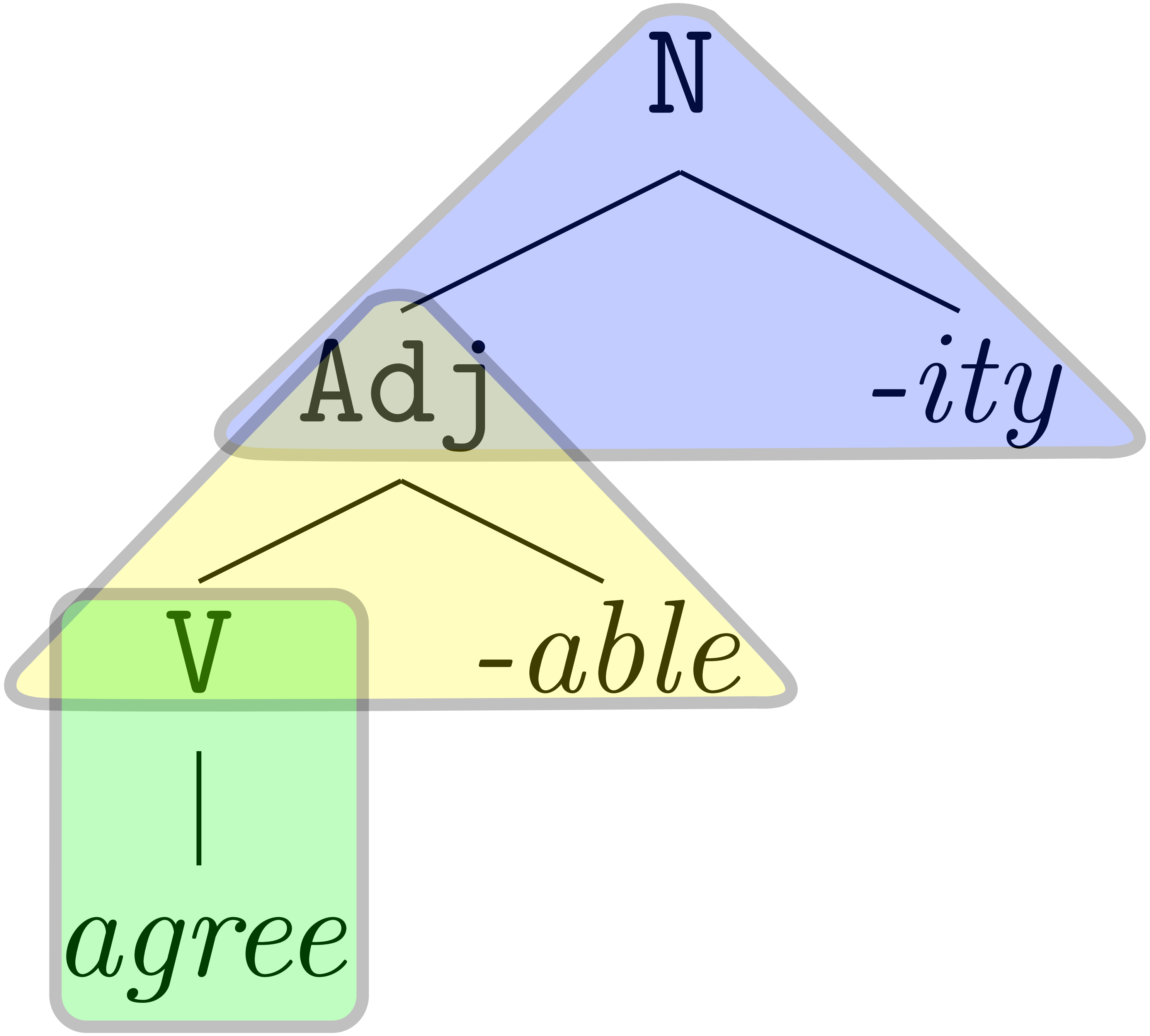
Hierarchical Structure

- Derivational morphology hierarchical and recursive.
    - Multiple suffixes can appear in a word.

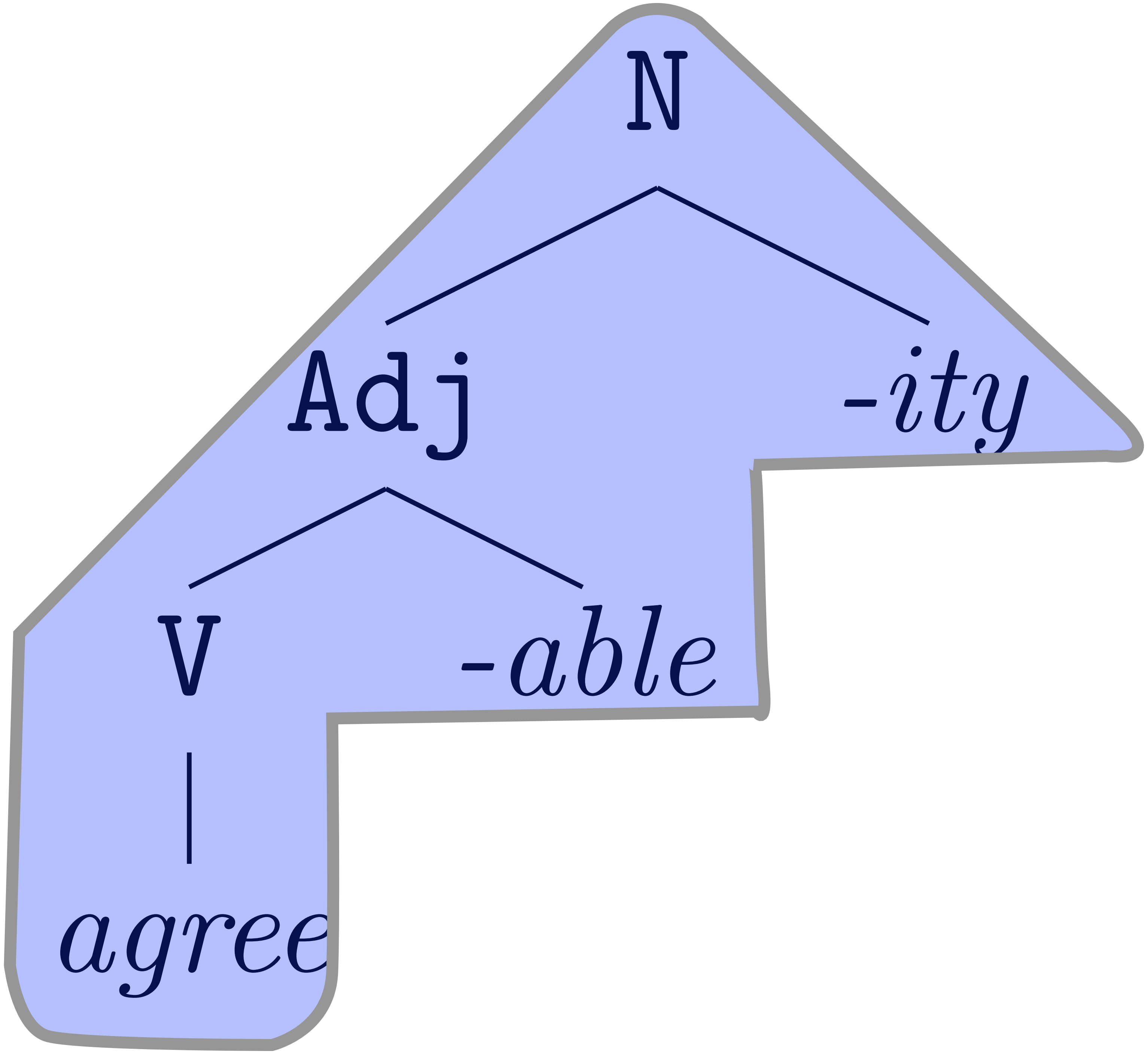




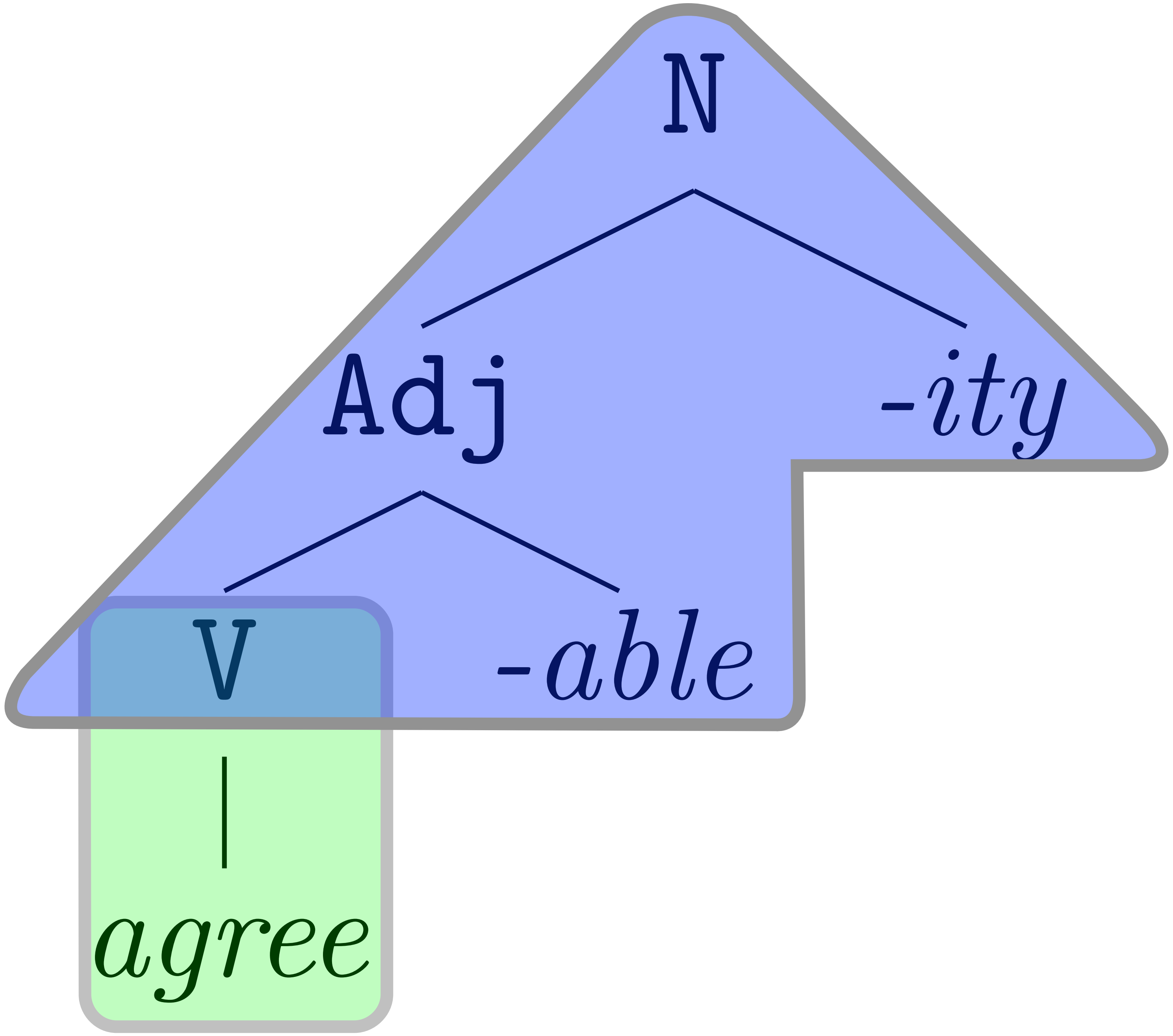
Many Hypotheses



Many Hypotheses



Many Hypotheses



Empirical Problem: Suffix Ordering

- Many combinations of suffixes do not appear in words.
- Fabb (1988).
    - 43 suffixes.
    - 663 possible pairs (taking into account selectional restrictions)
    - Only 50 exist.

Empirical Problem: Suffix Ordering

- Many theories
  - Level-ordering (e.g., Siegel, 1974)
  - Selectional-restriction based (e.g., Plag, 2003)
  - Complexity-based ordering (Hay, 2004)
- Focus on two phenomena
  - *Productivity and ordering generalization*
  - *Paradoxical suffix combinations*

Productivity and Ordering Generalization

(Hay, 2004)

On average, more productive suffixes appear after less productive suffixes

(Hay, 2002; Hay and Plag, 2004; Plag et al, 2009).

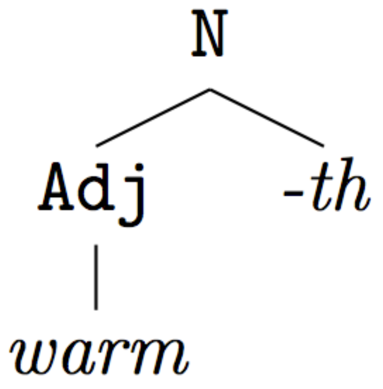
Productivity and Ordering Generalization

(Hay, 2004)

- Implicit in many earlier theories (e.g., Level-Ordering Generalization of Siegel 1974).
- Hay's argues for processing-based view *(Complexity-Based Ordering)*
- But: Follows as a logically necessary consequence of pattern of storage and computation.

Productivity and Ordering Generalization

- Intuition:
  - Less productive suffixes stored as part of words.



  - More productive suffixes can attach to anything, including morphologically-complex stored forms.

But: Paradoxical Suffix Combinations

- Combinations of suffixes which violate the Productivity and Ordering Generalization (as well as predictions of other earlier theories).
  - *-ability*, *-ation*, *-istic*, *-mental*

Multi-way Competition: *-ity* v. *-ness*

- In general, *-ness* more productive than *-ity*.
- *-ity* more productive after:

  -ile, *-able*, *-(i)an*, *-ic*.

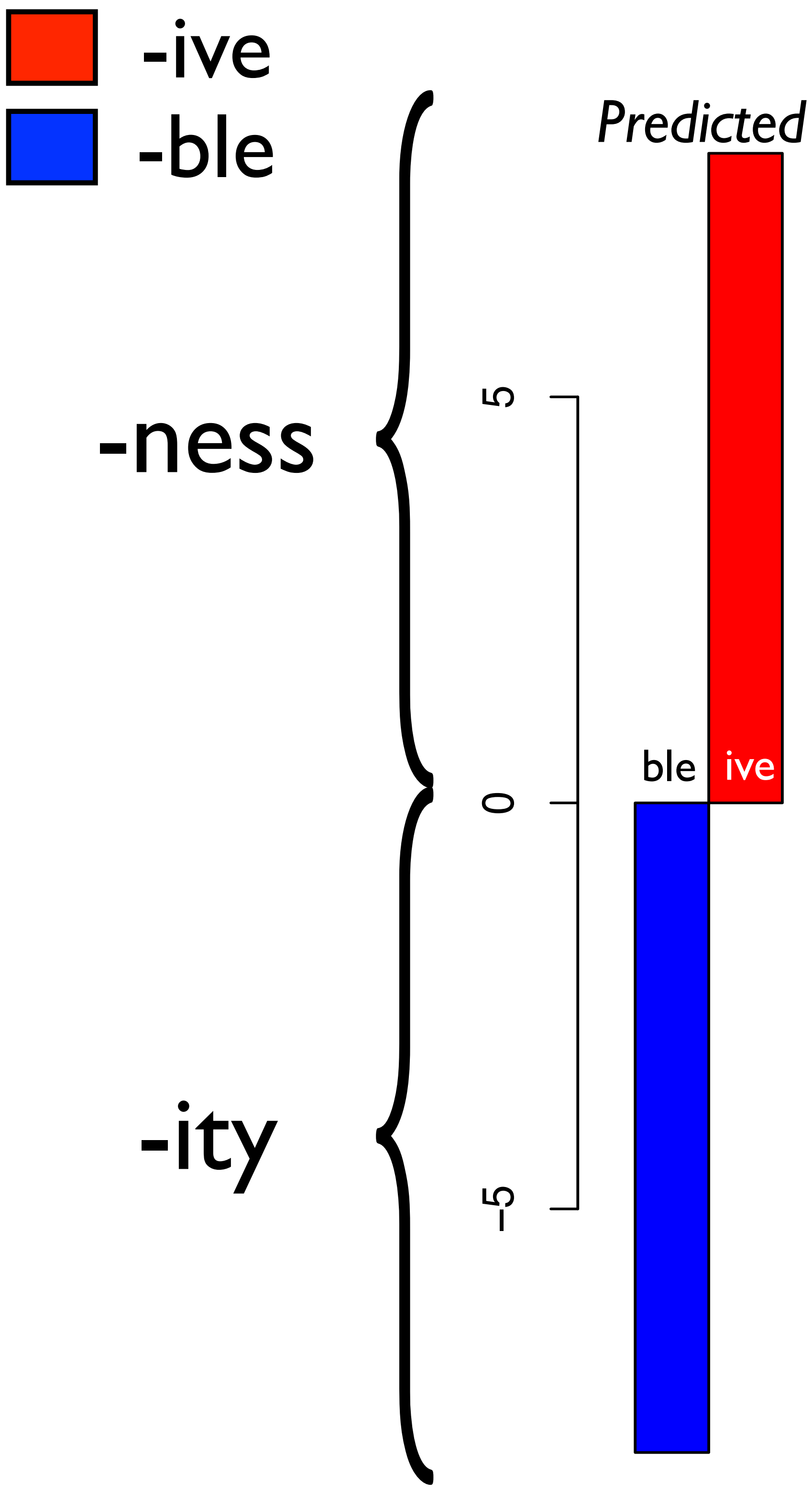
(Anshen & Aronoff, 1981; Aronoff & Schvaneveldt, 1978; Cutler, 1980)

Two Frequent Combinations: -ivity v. -bility

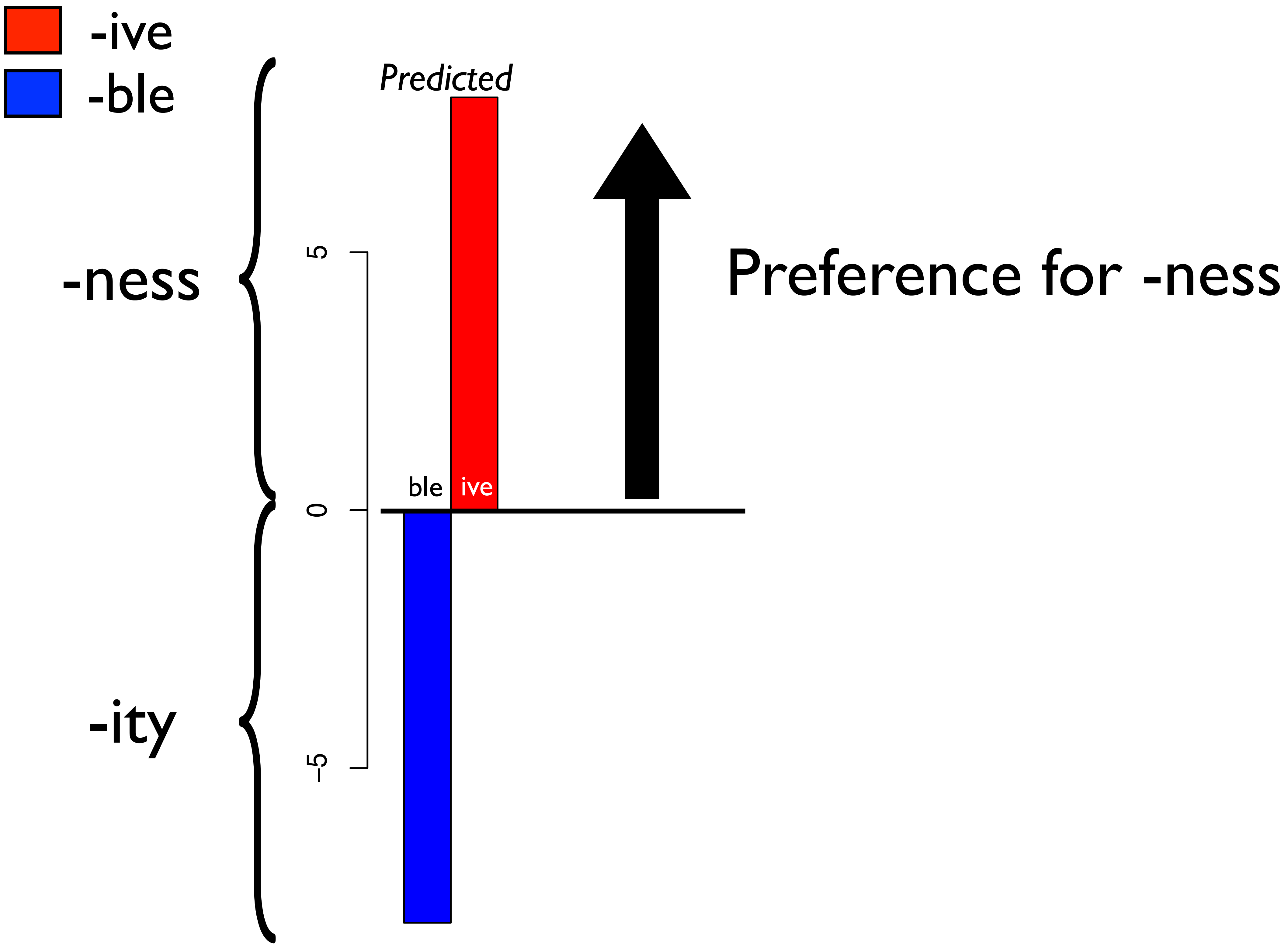
- *-ive* + *-ity*: ***-ivity*** (e.g., *selectivity*).
  - Speaker prefer to use -ness with novel words (Aronoff & Schvaneveldt, 1978).
  - *depulsiveness* > *depulsivity*.

- *-ble* + *-ity*: ***-bility*** (e.g., *sensibility*).
  - Speakers prefer to use -ity with novel words (Anshen & Aronoff, 1981).
  - *remortibility* > *remortibleness*.

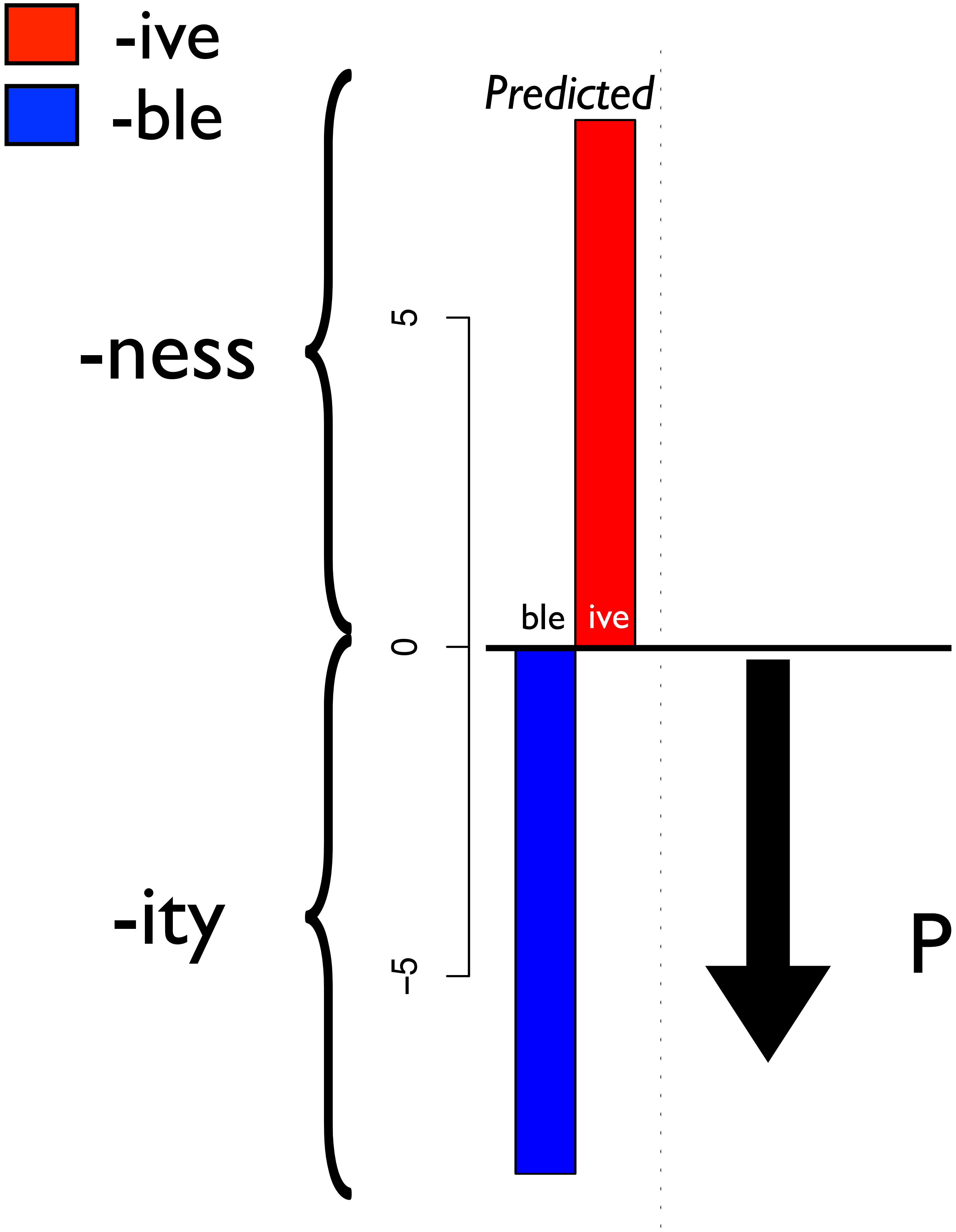
-ivity v. -bility



-ivity v. -bility

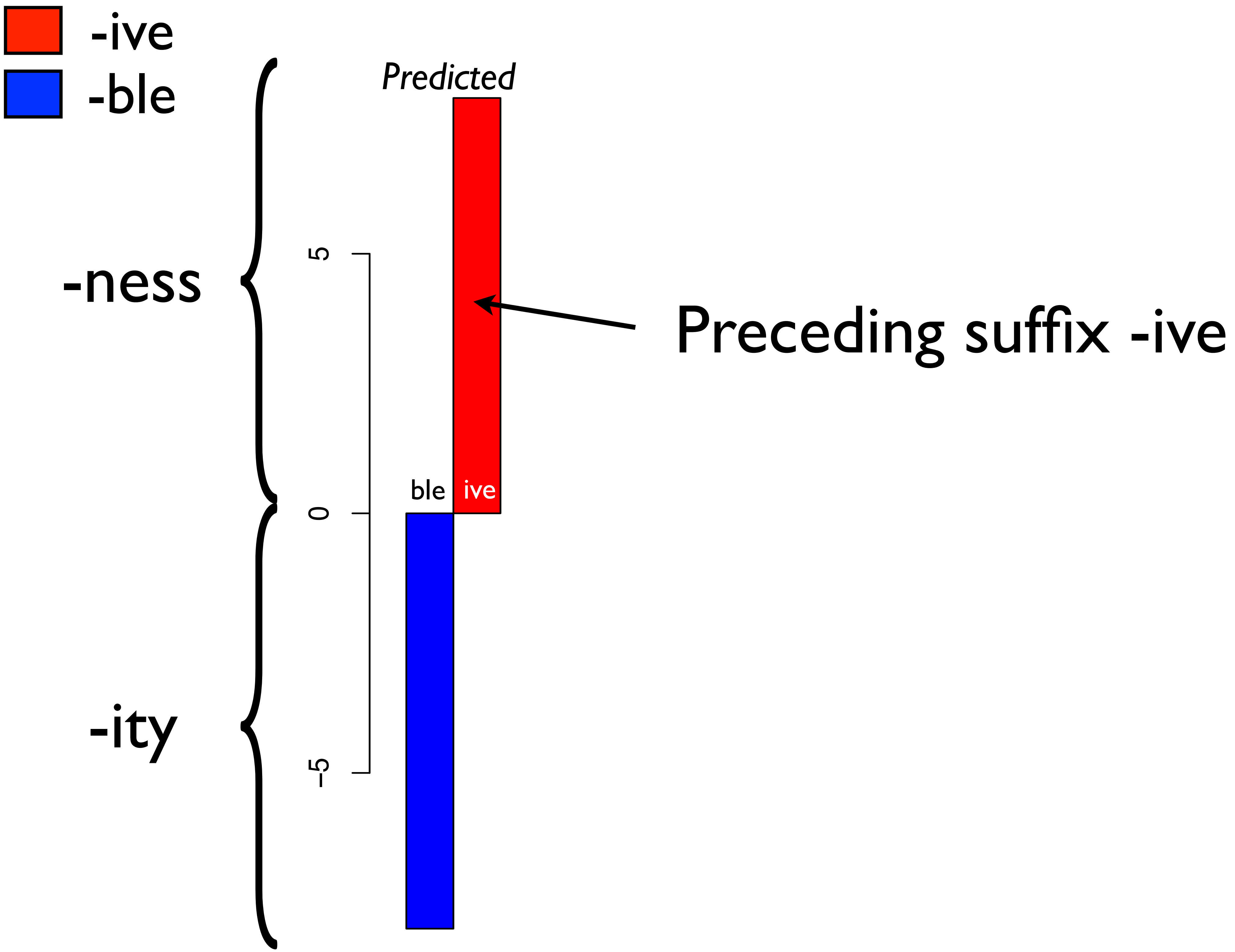


-ivity v. -bility



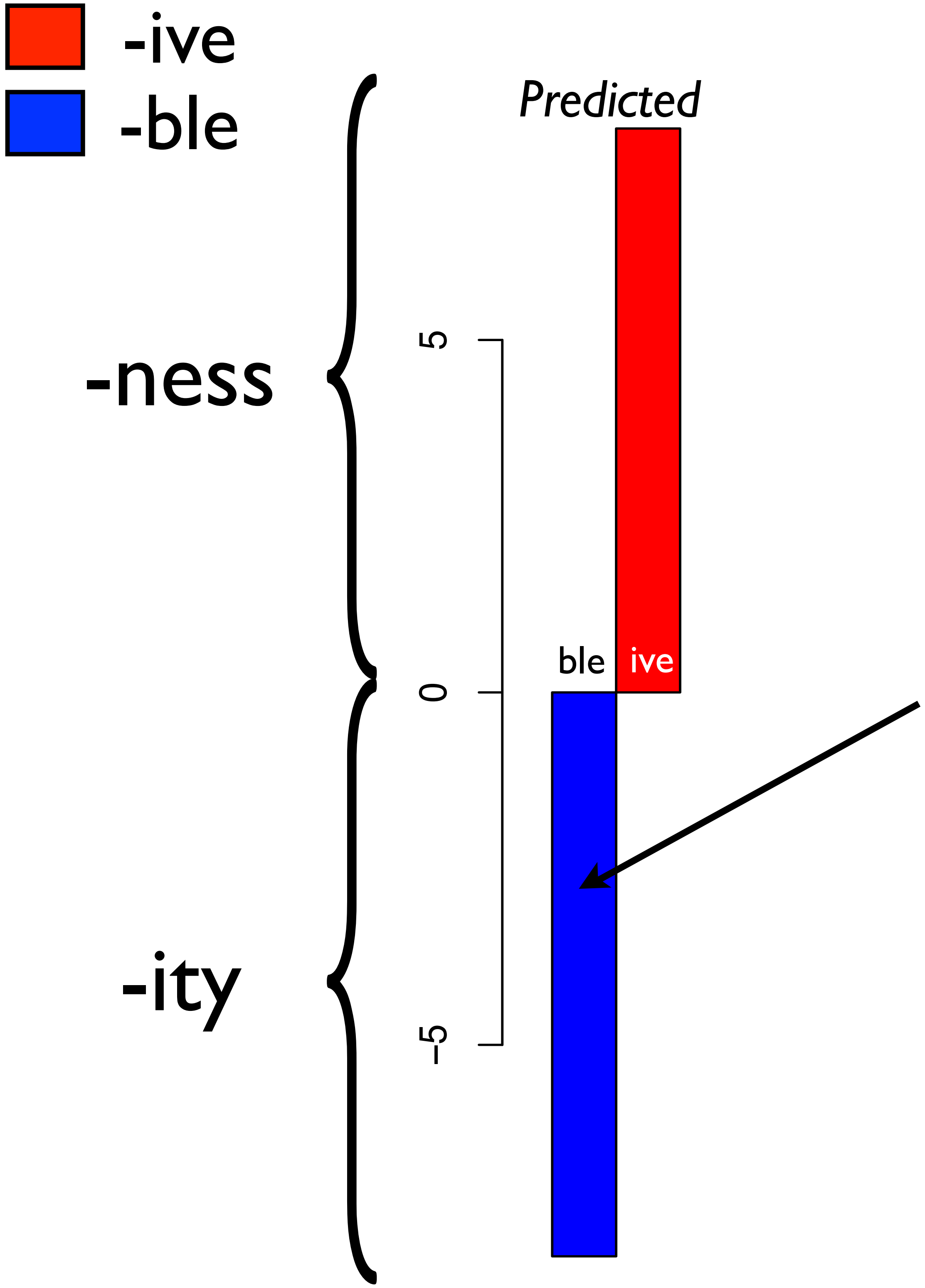
Preference for -ity

-ivity v. -bility



Preceding suffix -ive

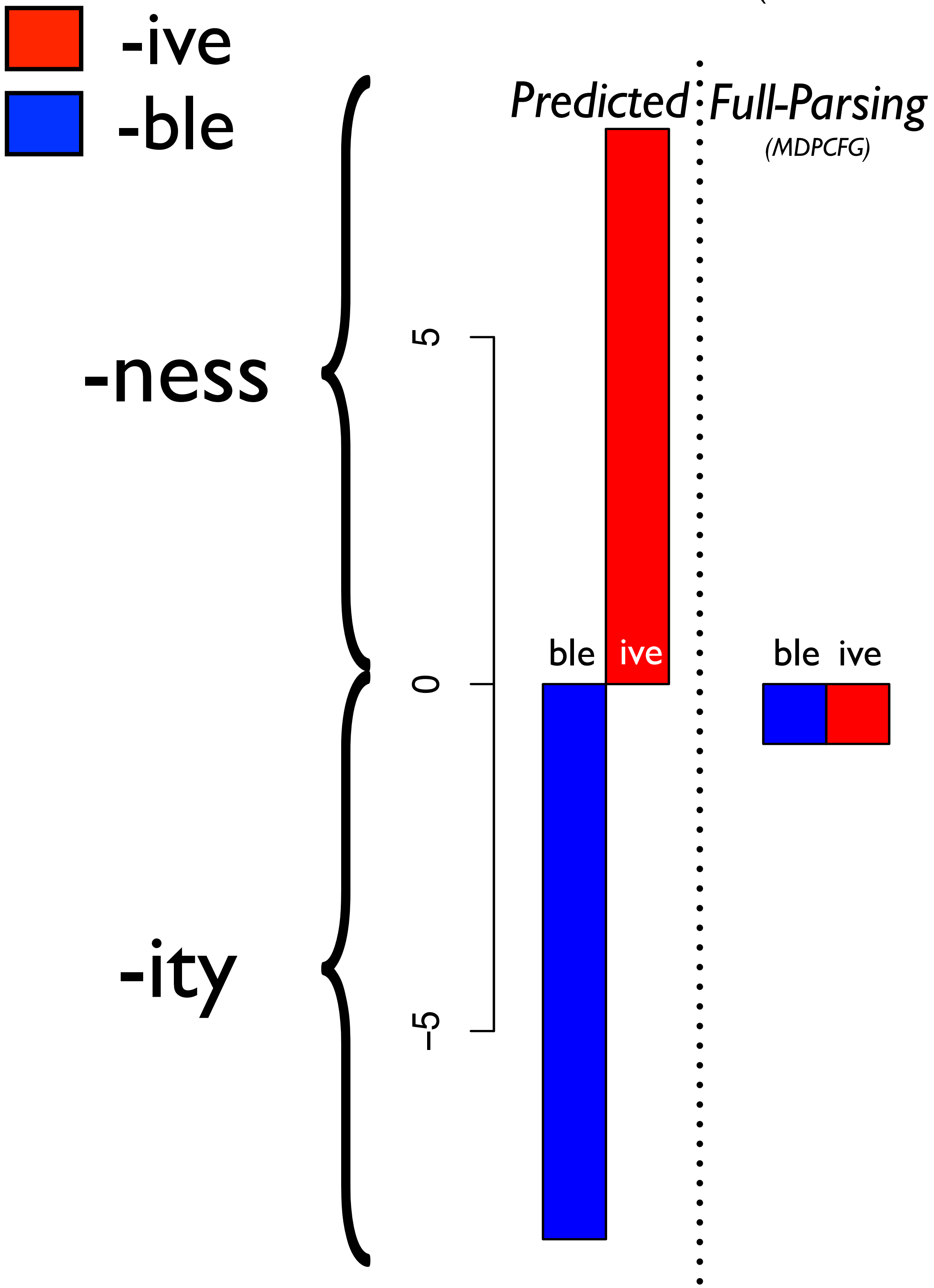
-ivity v. -bility



Preceding suffix -ble

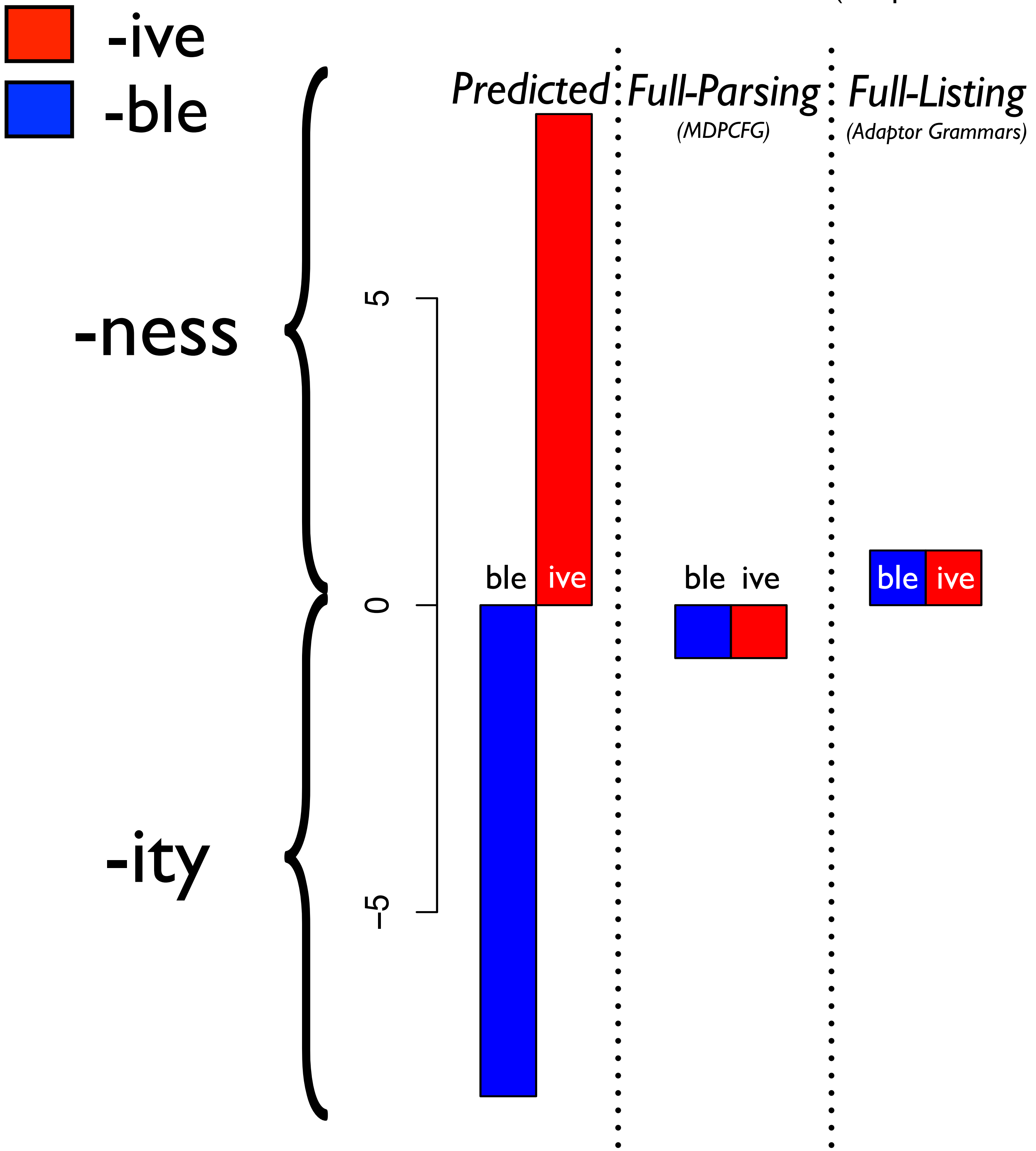
Full-Parsing

(Multinomial-Dirichlet Context-Free Grammar)



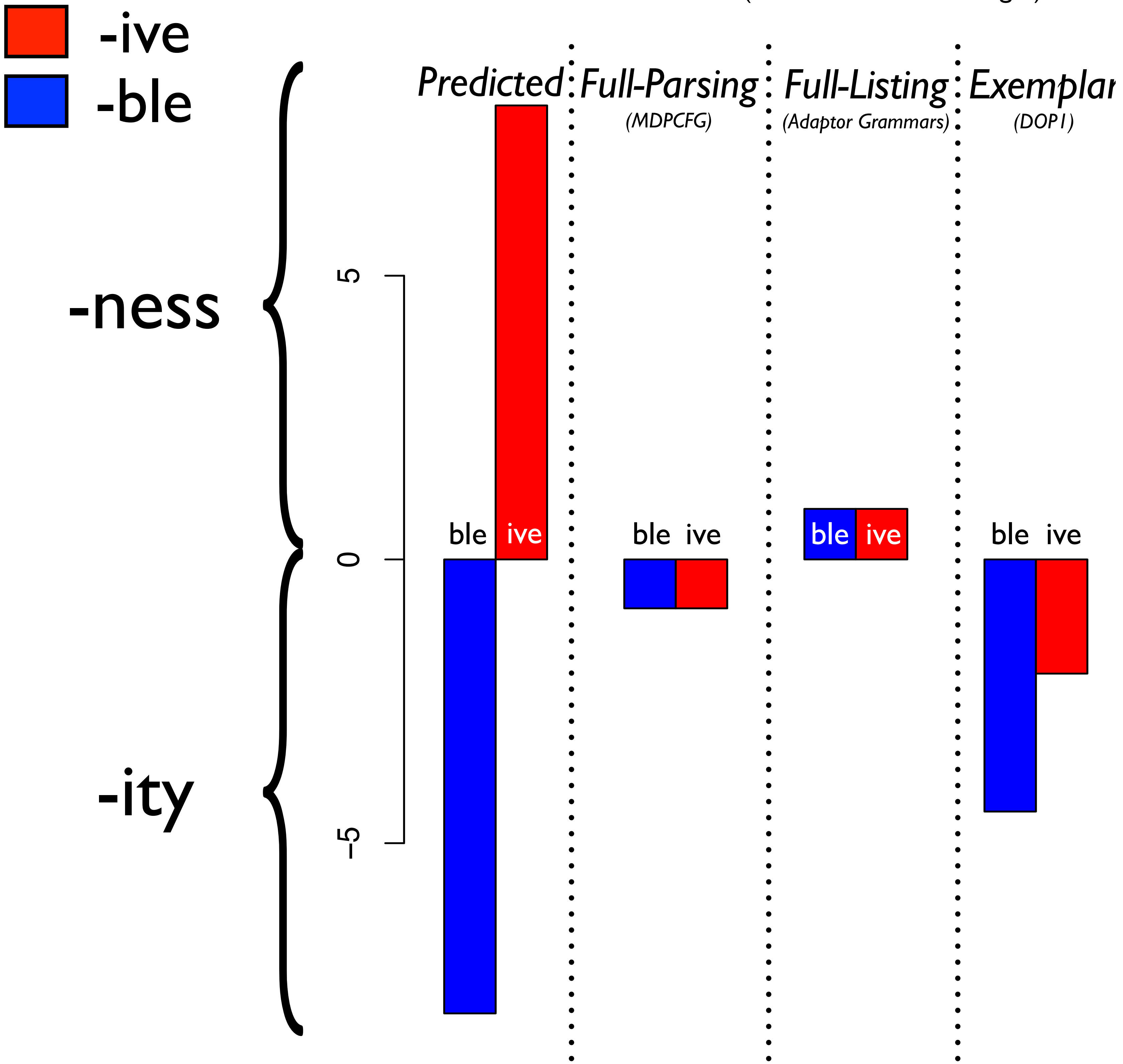
Full-Listing

(Adaptor Grammars)



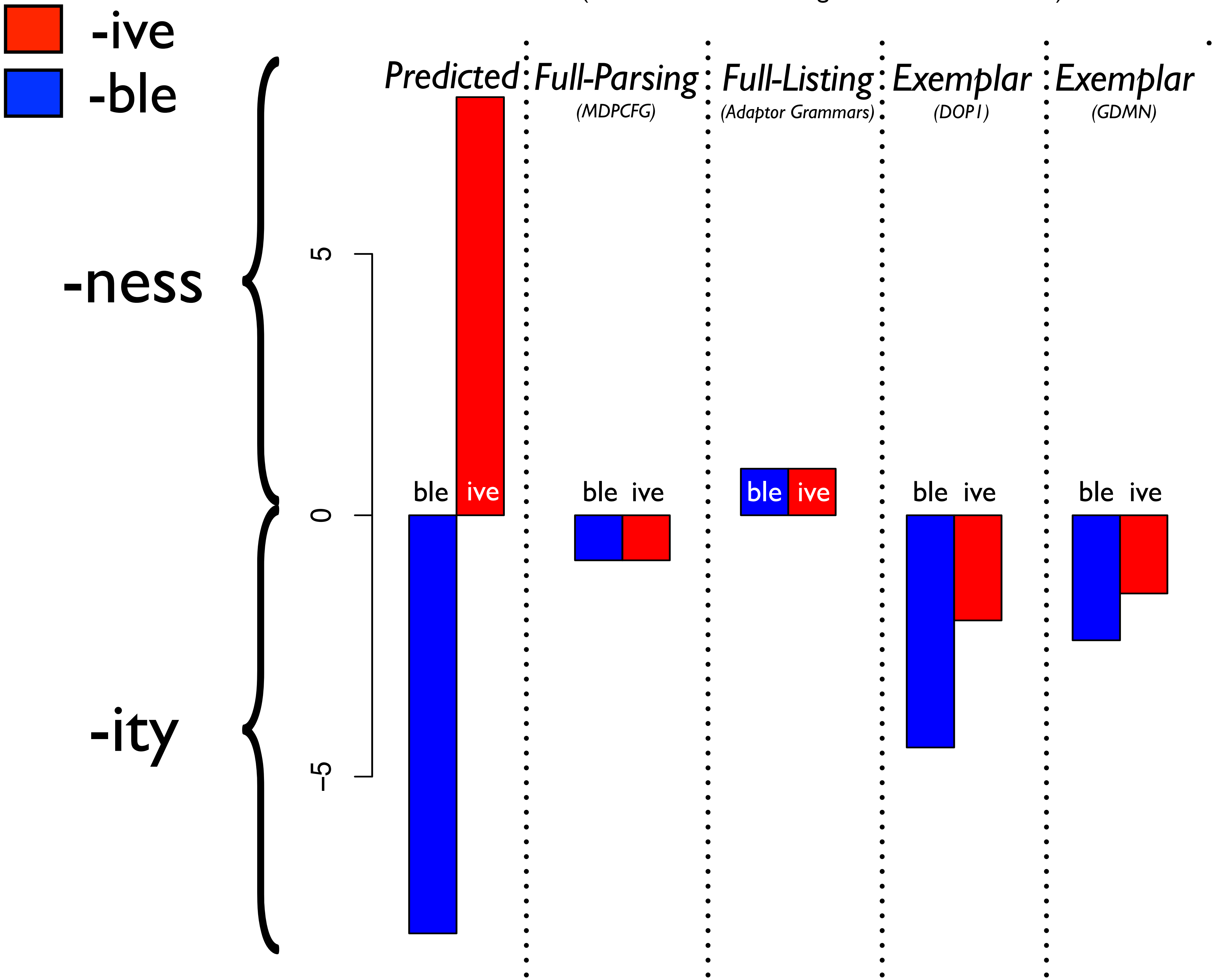
Exemplar-Based

(Data-Oriented Parsing 1)



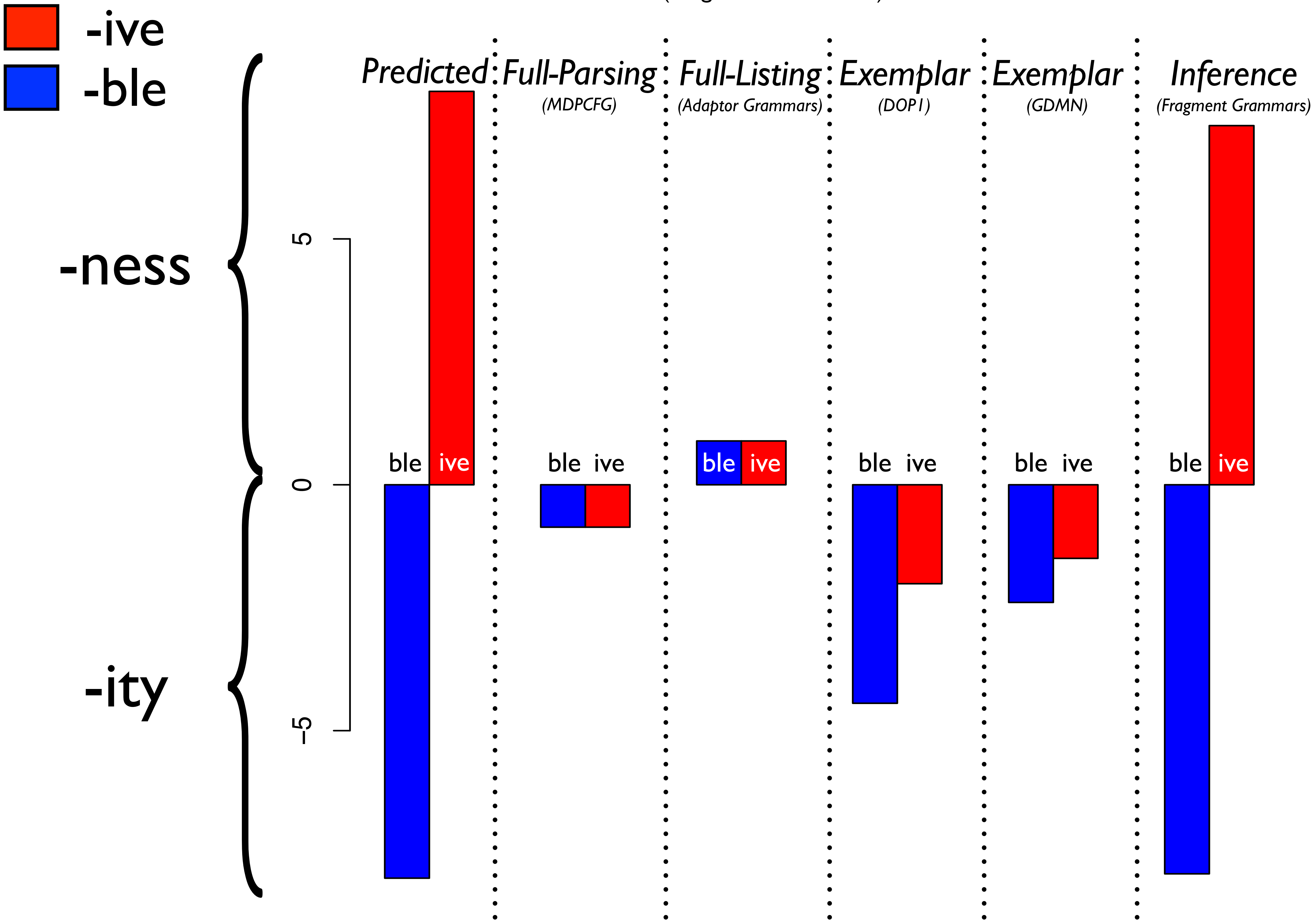
Exemplar-Based

(Data-Oriented Parsing: Goodman Estimator)



Inference-Based

(Fragment Grammars)



Multi-way Competition

- Explains *productivity and ordering generalization.*
- Explains difficult cases of competition involving *paradoxical suffix combinations*.

Global Summary

- Inference based on distribution of tokens over types.
  - Derives Baayen's *hapax*-based theory.
- View the choice of whether to retrieve or compute as an inference.
  - Derives e*lsewhere condition*.
- Storage of arbitrary structures explains ordering generalizations.
  - Explains *Productivity and Ordering Generalization*.
  - Also accounts for *paradoxical suffix combinations* such as *-ability*

Conclusion

- Model the problem of deriving word forms using a mixture of computation and storage as a tradeoff using standard inferential tools.
- Automatically solves many problems of productivity and competition resolution.

Thanks!