Hierarchical Search for Parsing

Adam Pauls and Dan Klein

Motivation

- Modern parsers users very large grammars (millions of rules!)

- Coarse-to-Fine has proven successful (Charniak and Caraballo 1998)

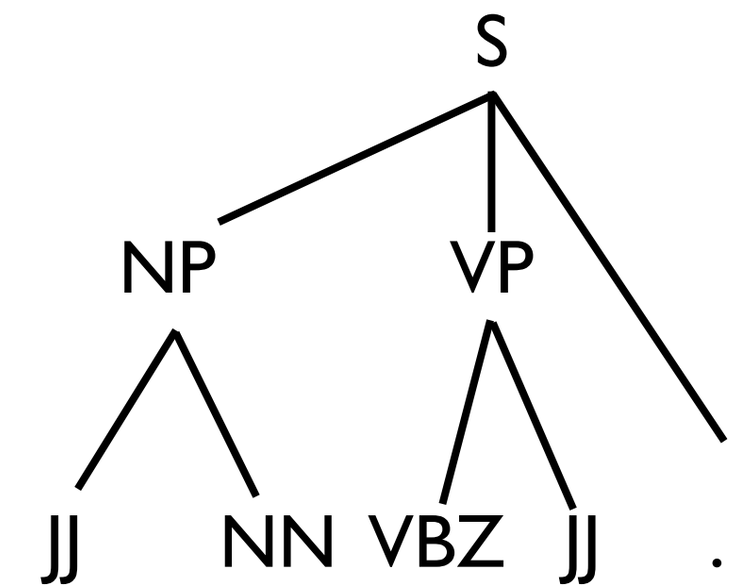
- Multi-level or Hierarchical Coarse-to-Fine works even better (Charniak and Johnson 2005, Petrov and Klein 2007)

- In this talk, we explore an optimal hierarchical search method: *Hierarchical A**

Hierarchical Setting

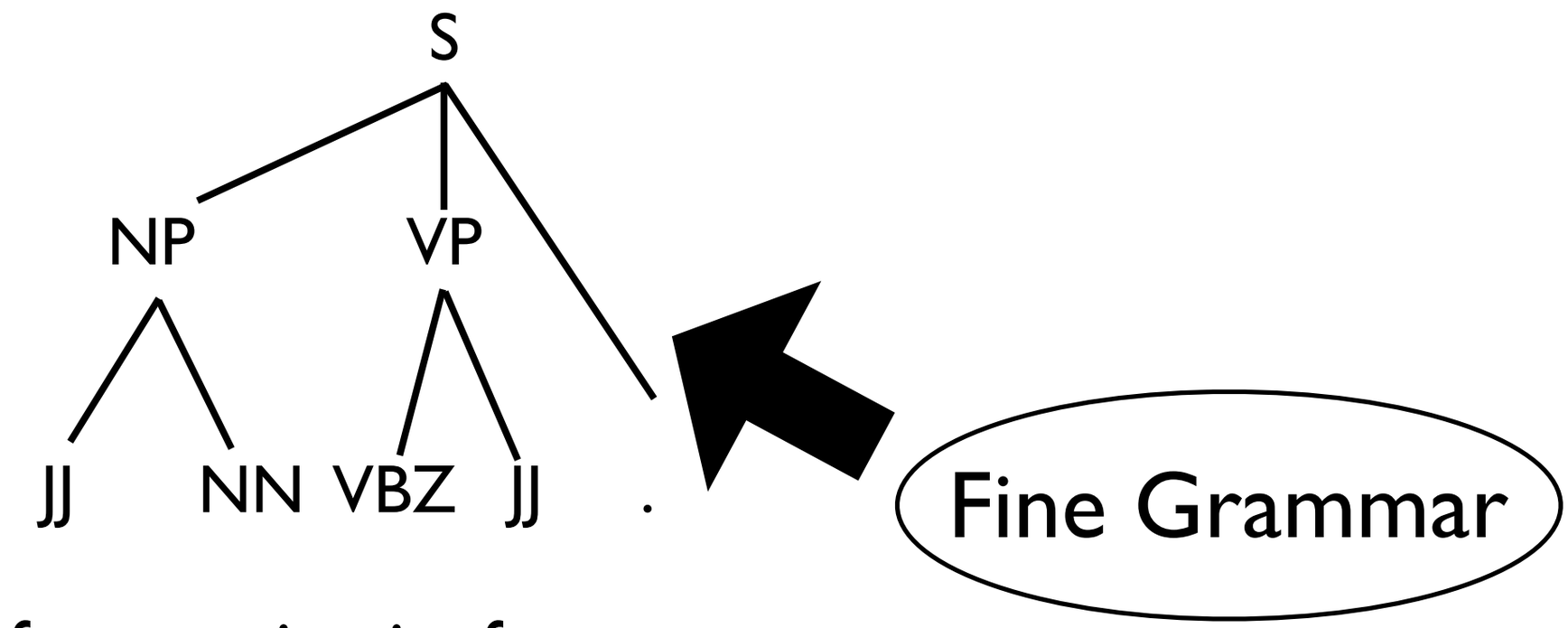
fast parsing is fun .

Hierarchical Setting

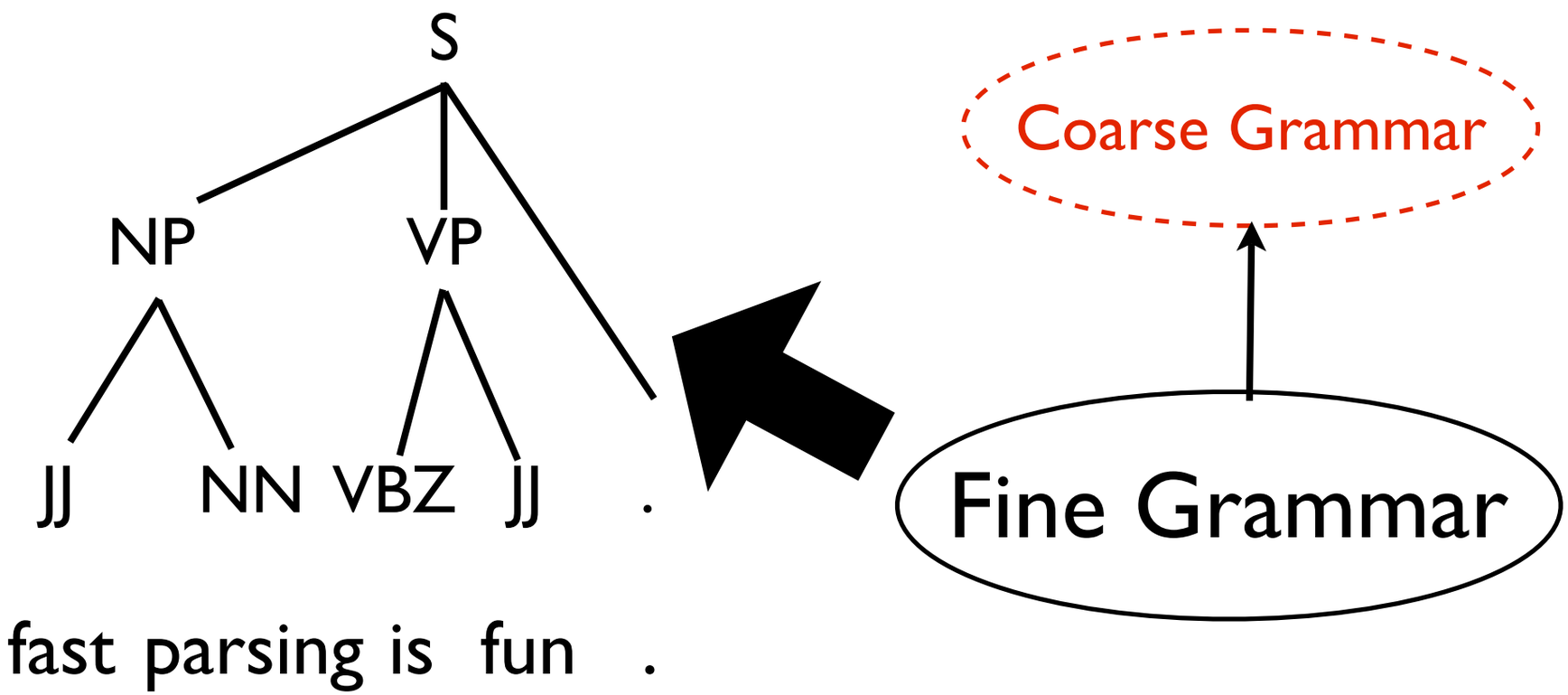


fast parsing is fun .

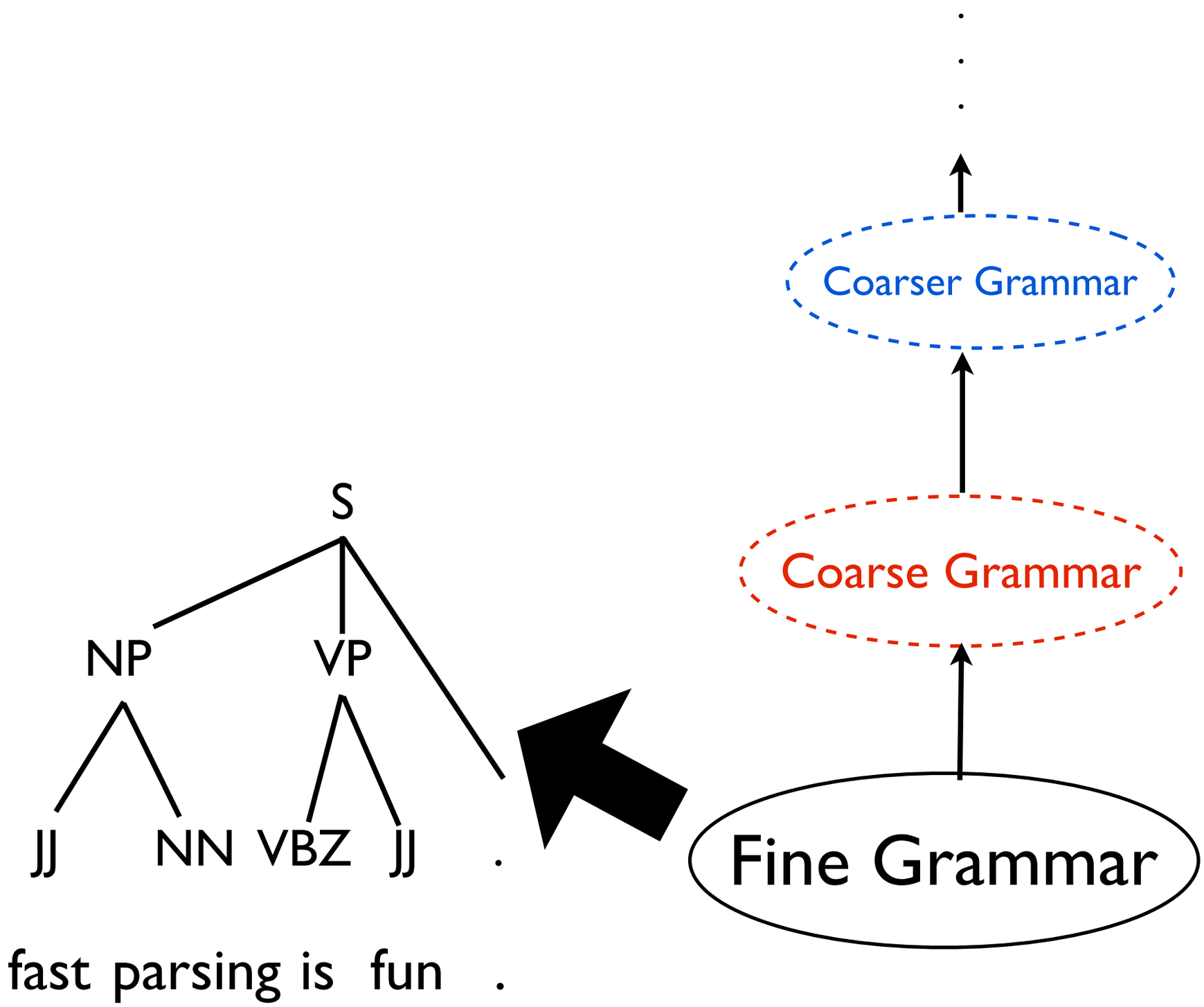
Hierarchical Setting



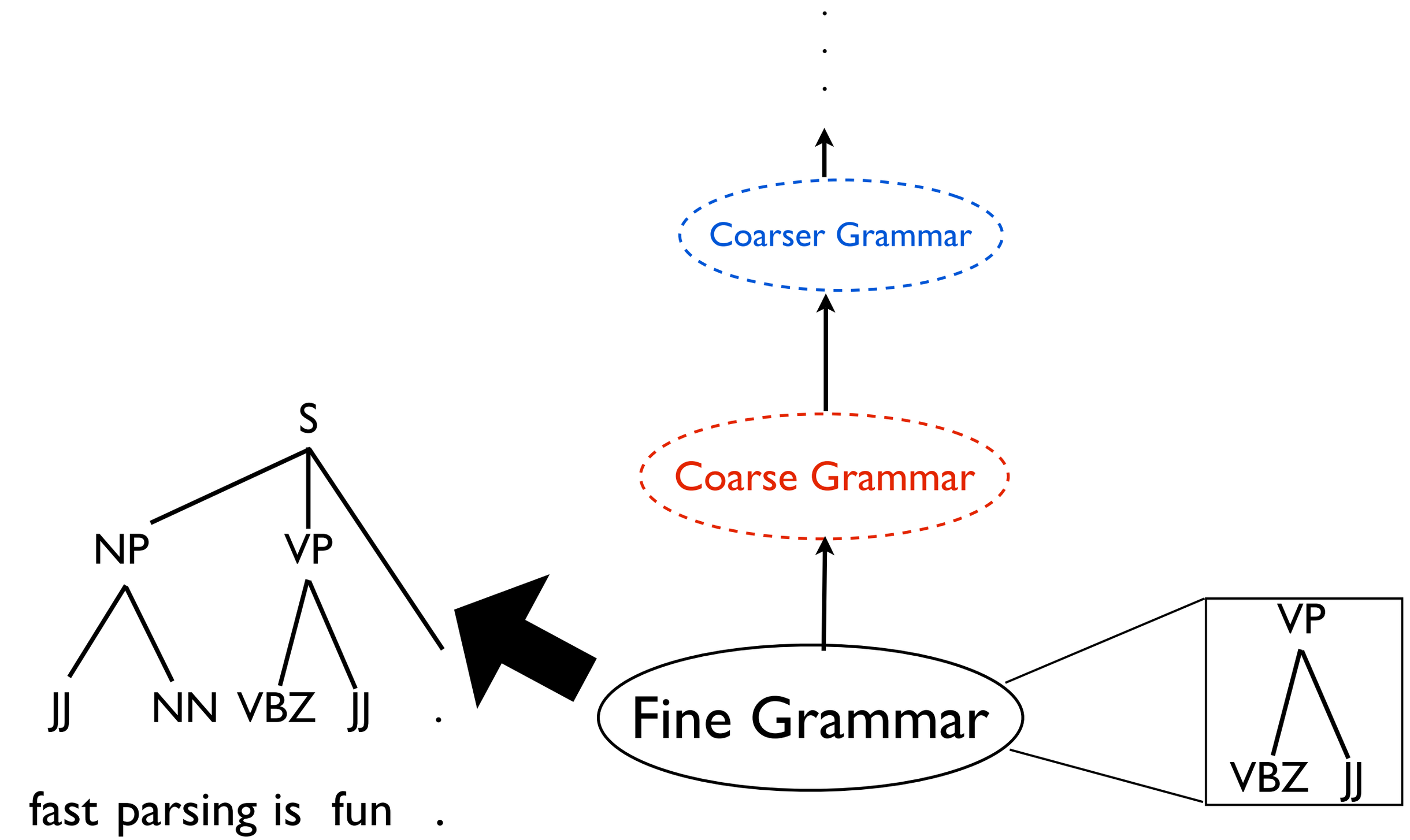
Hierarchical Setting



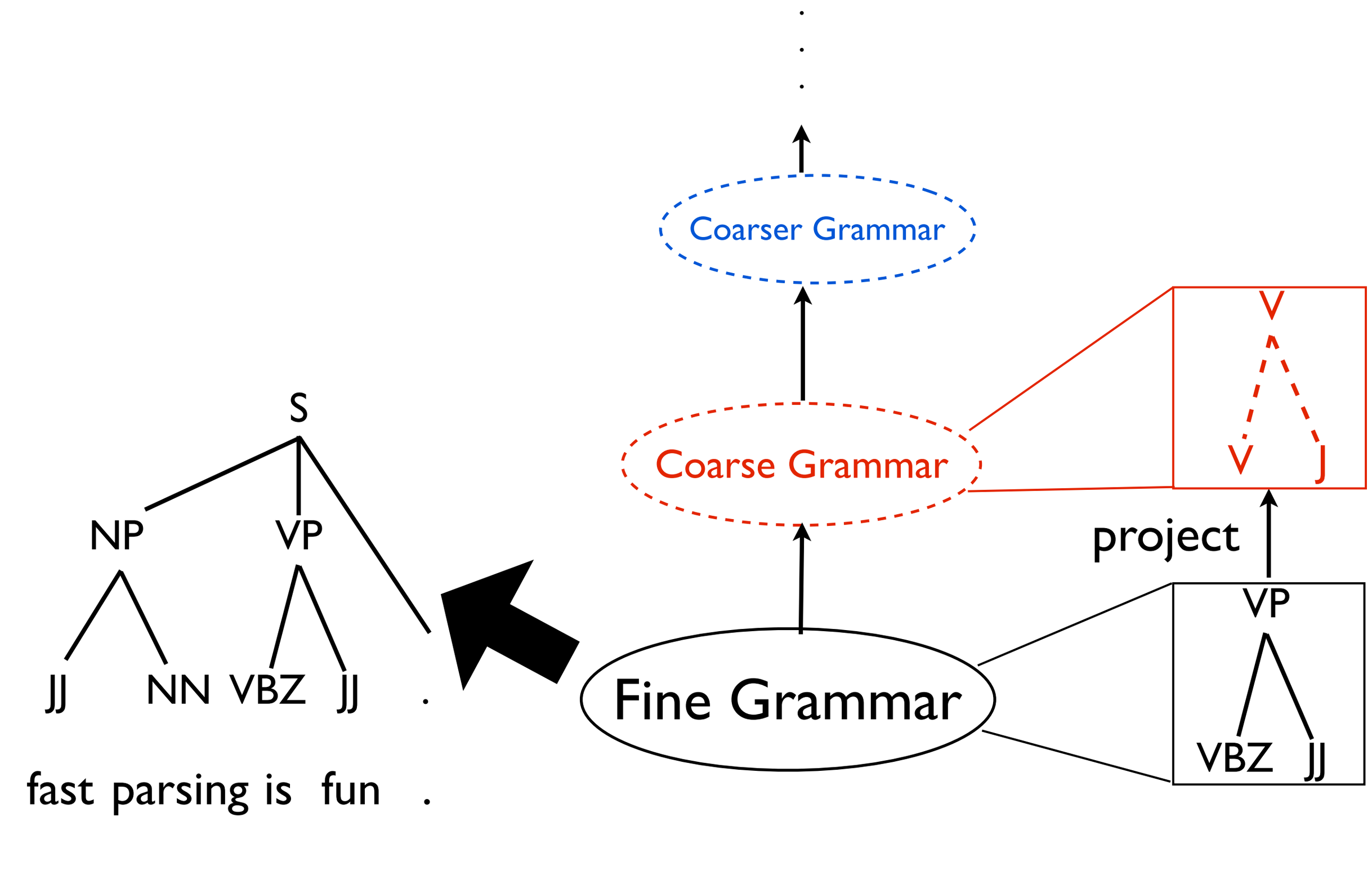
Hierarchical Setting



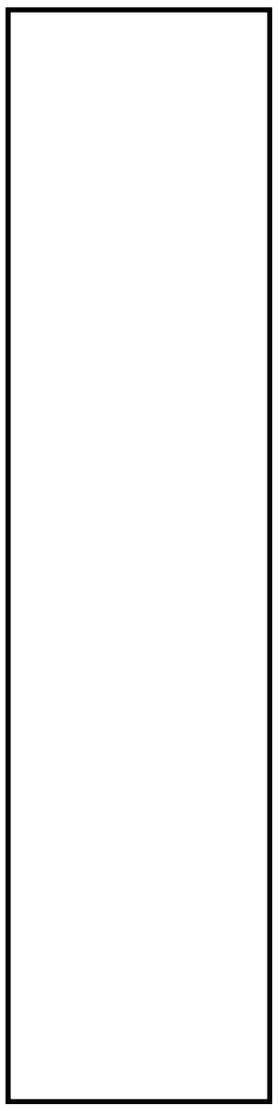
Hierarchical Setting



Hierarchical Setting

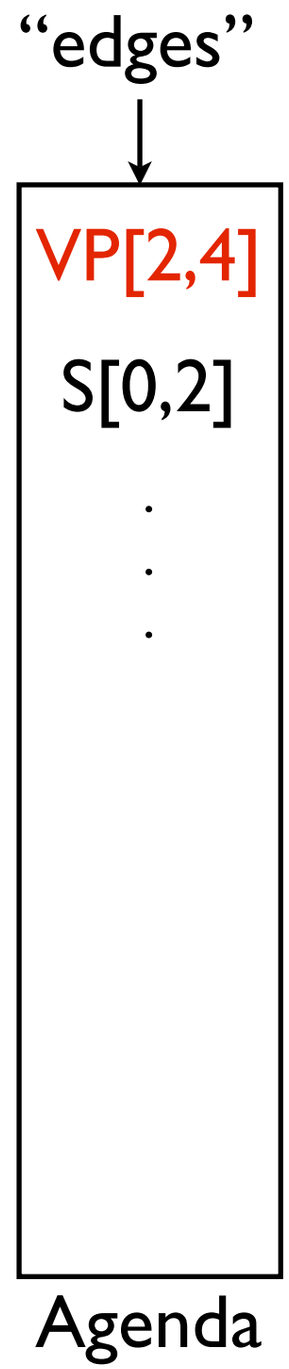


Agenda-Based Search

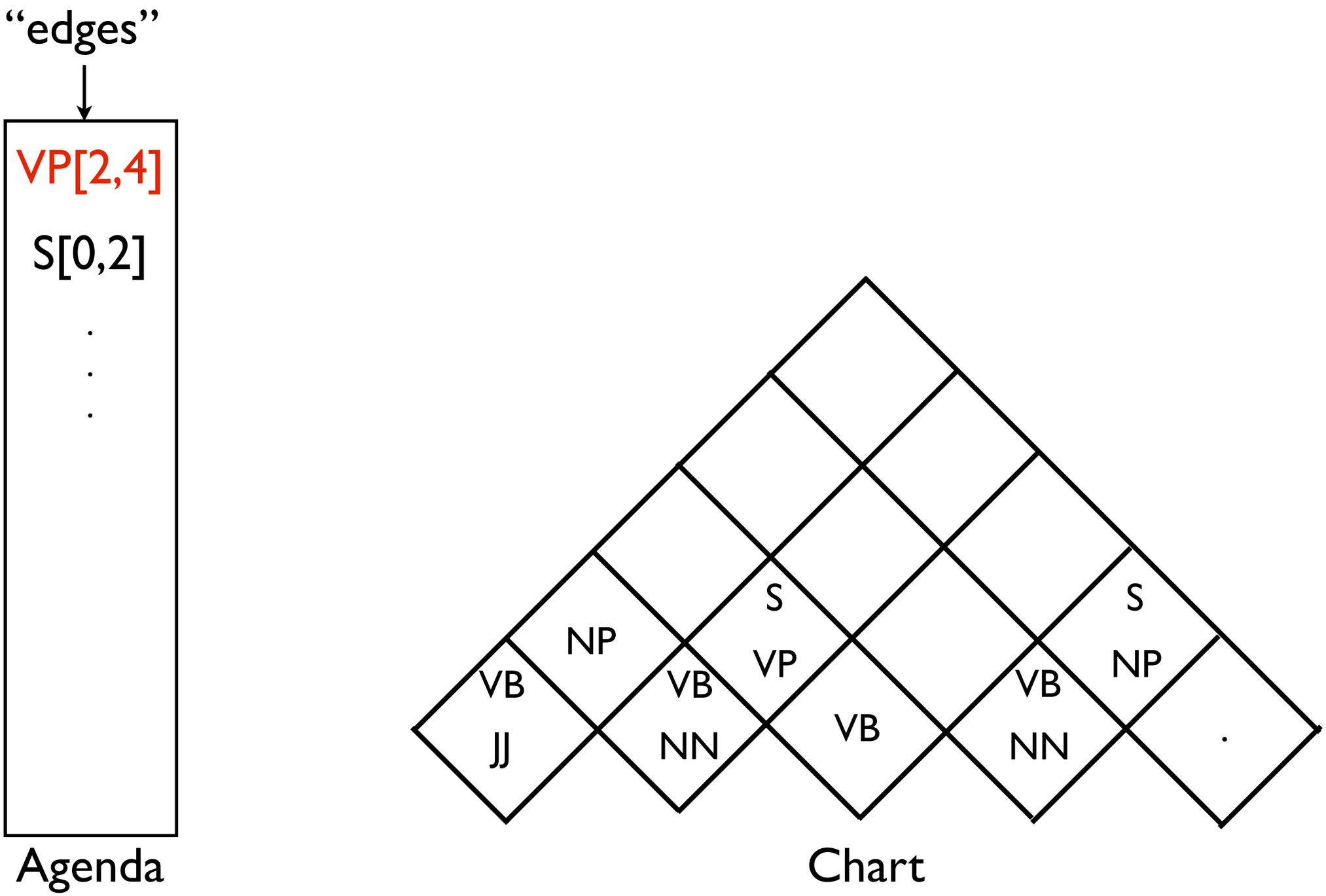


Agenda

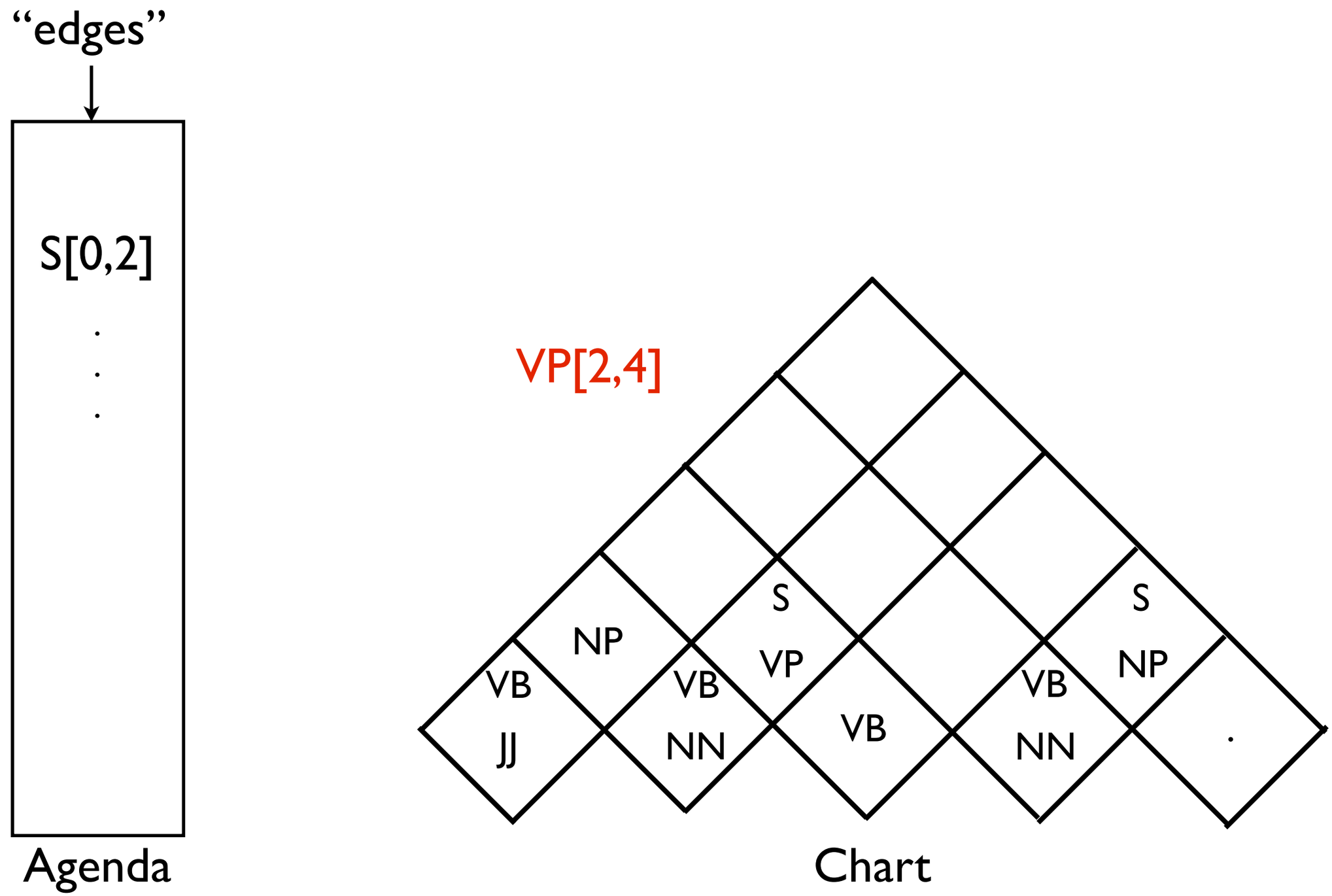
Agenda-Based Search



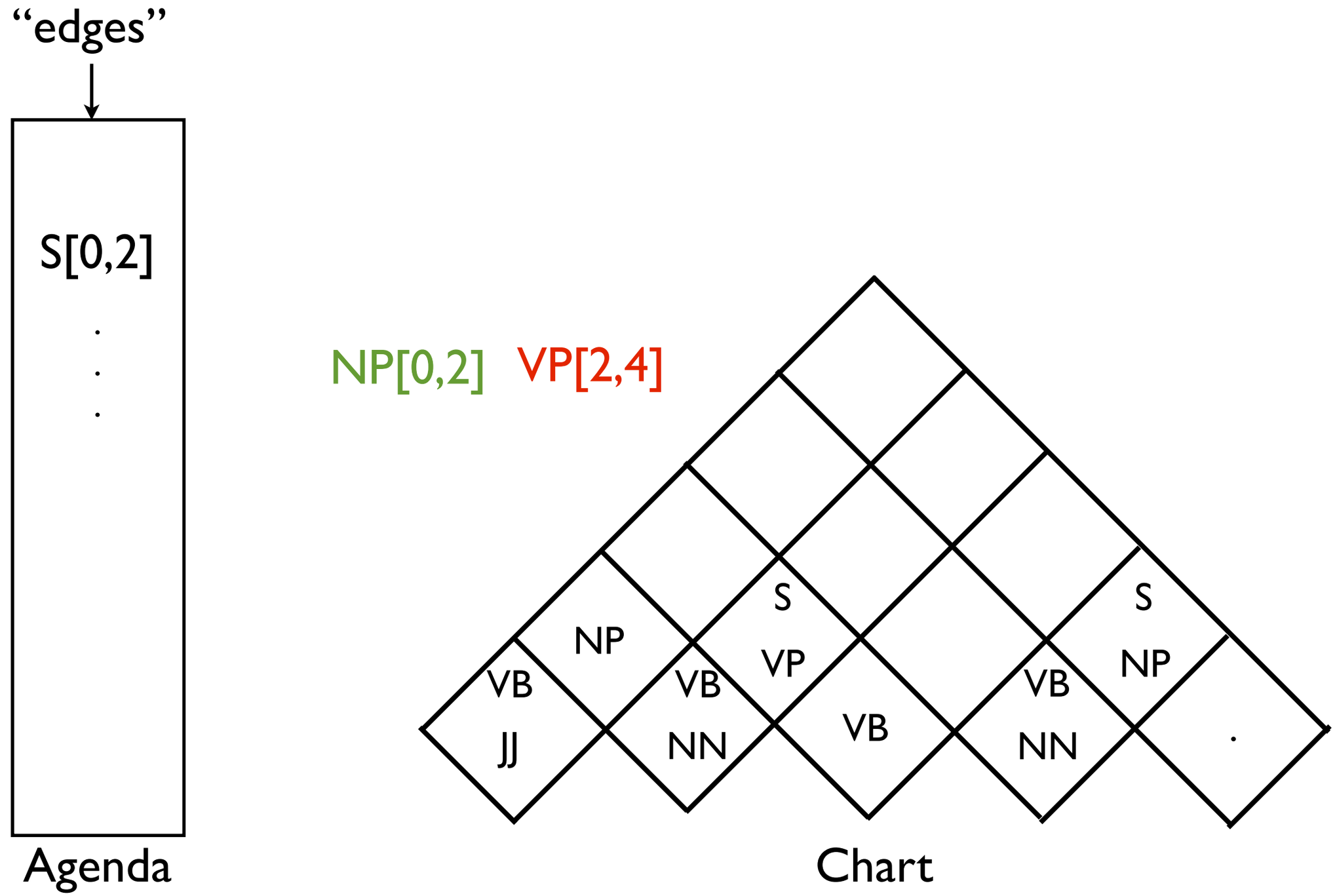
Agenda-Based Search



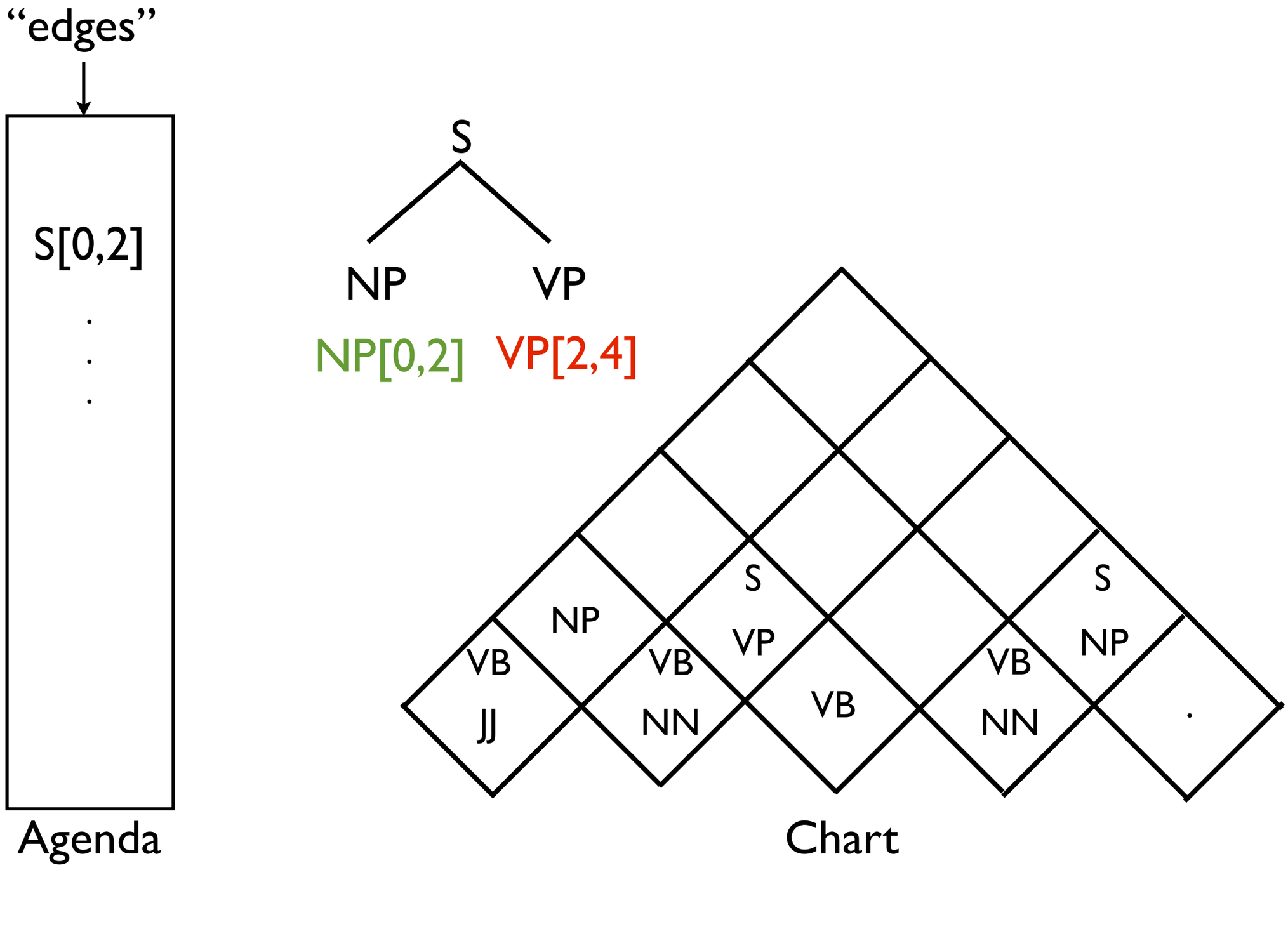
Agenda-Based Search



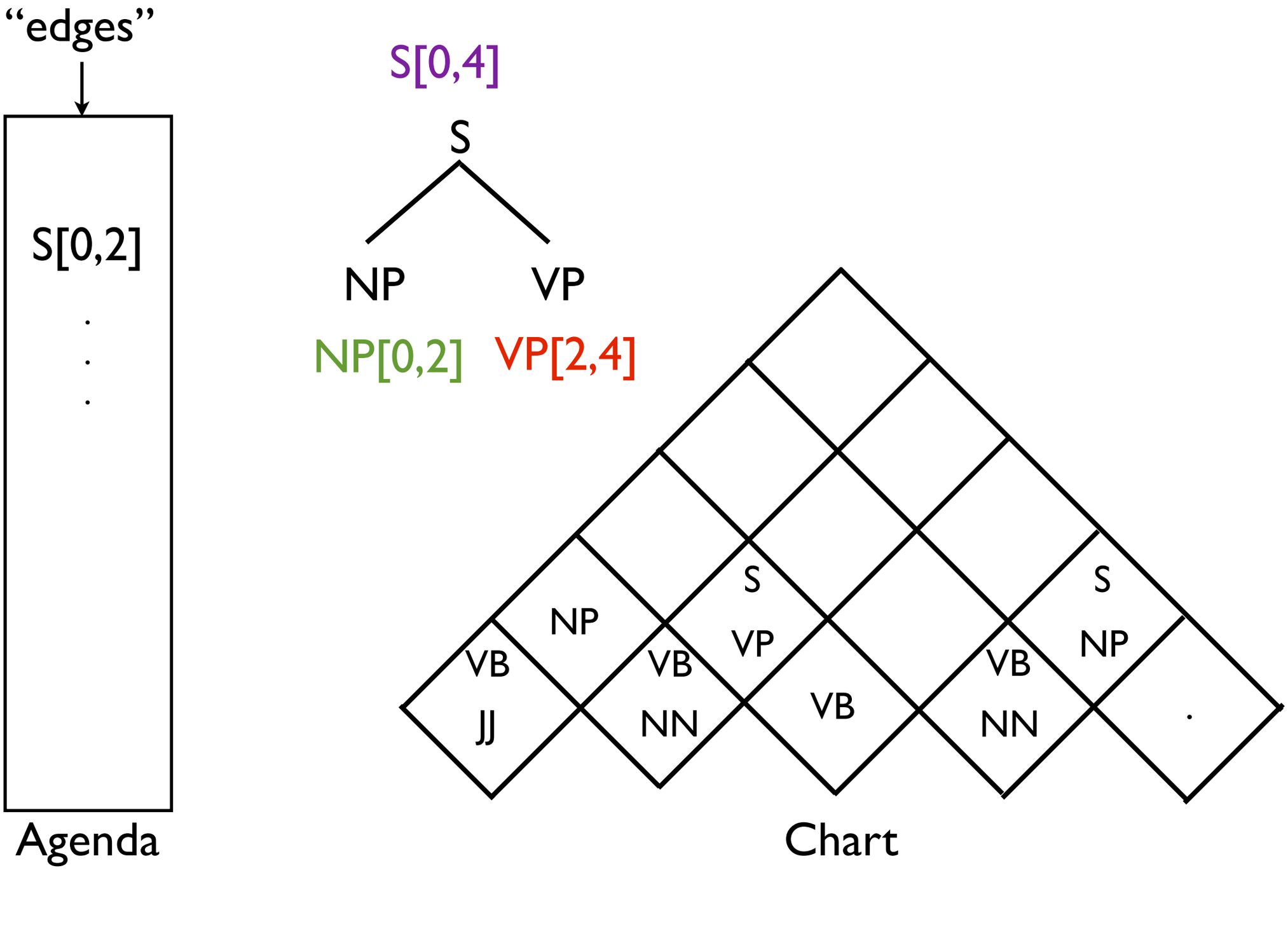
Agenda-Based Search



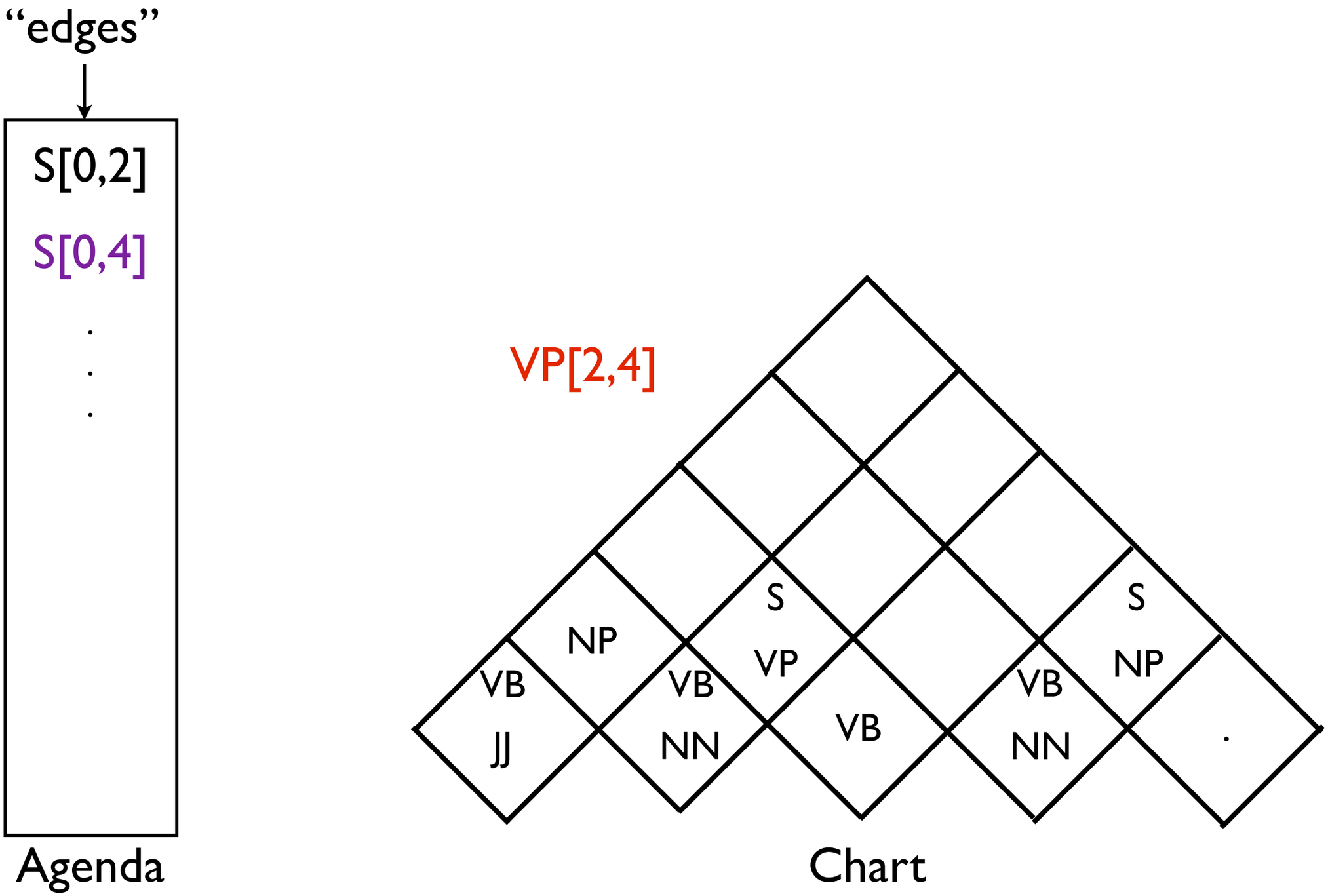
Agenda-Based Search



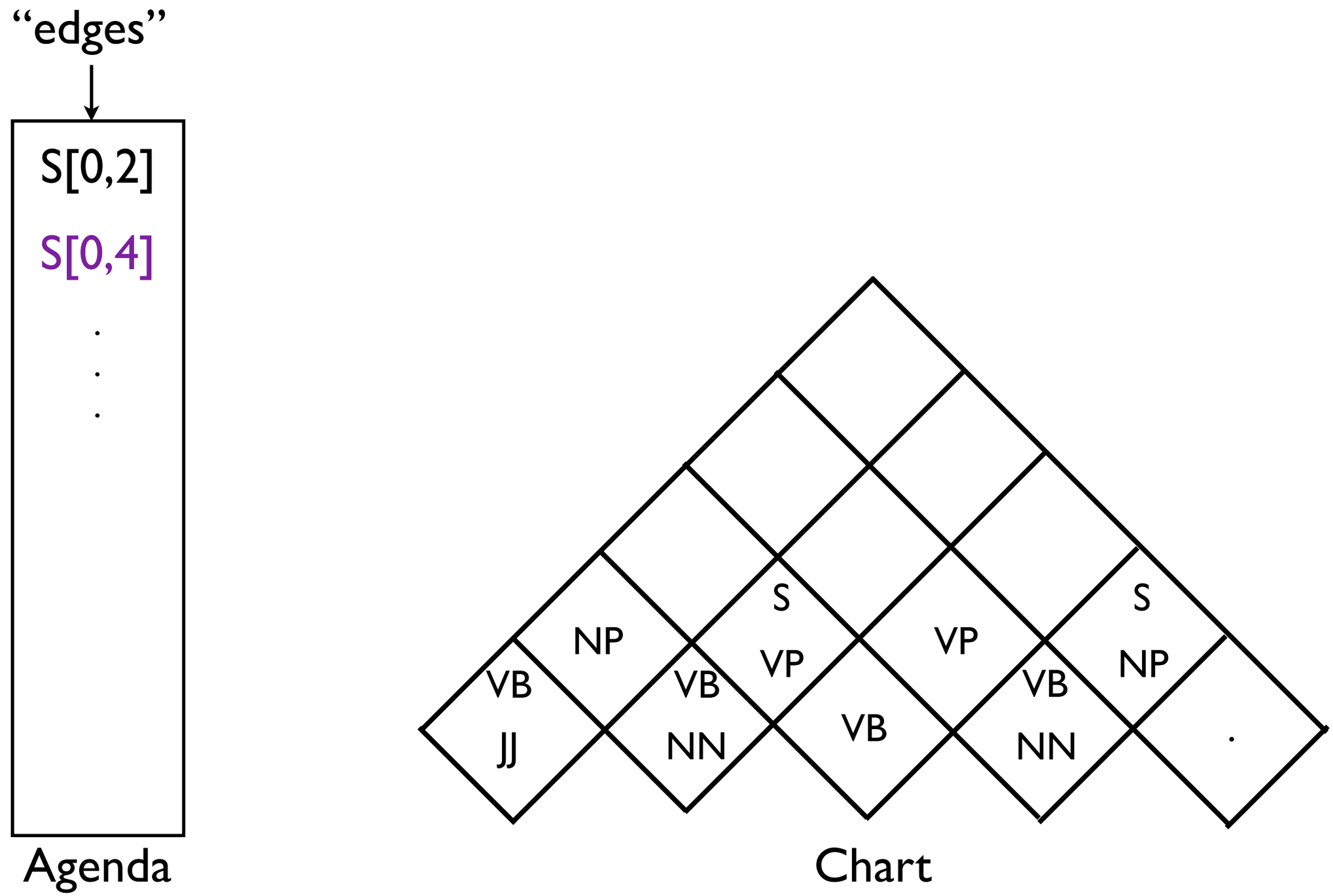
Agenda-Based Search



Agenda-Based Search



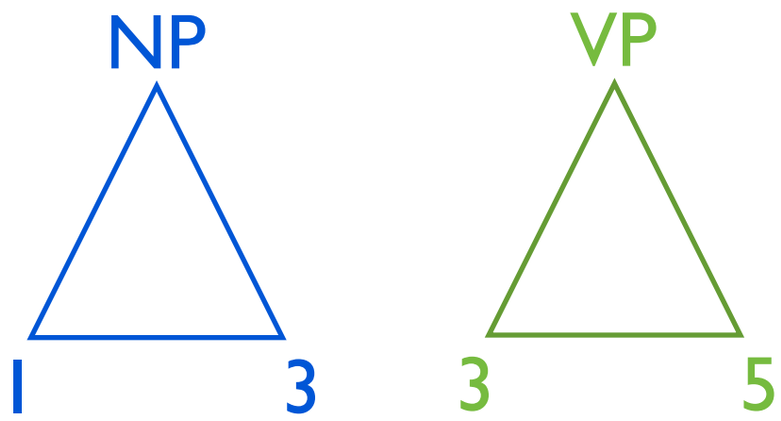
Agenda-Based Search



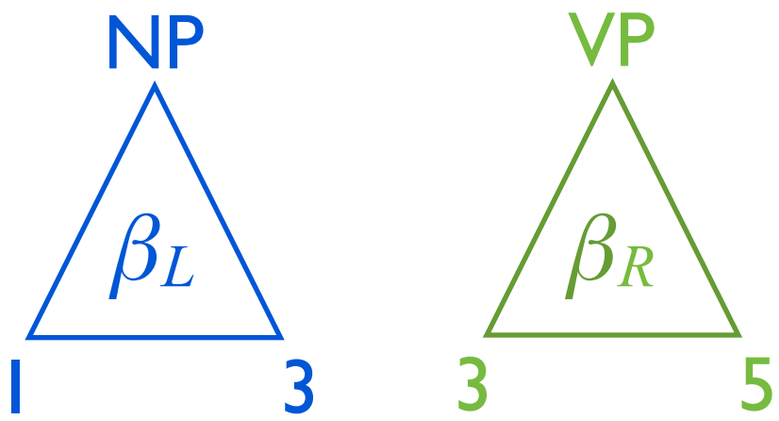
University of California
Berkeley

Building Edges

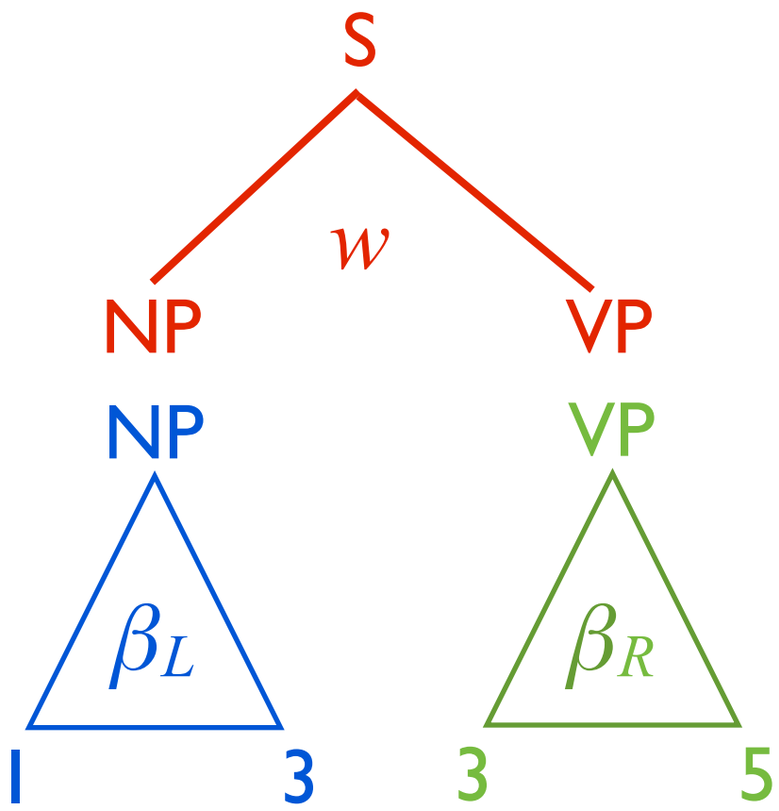
Building Edges



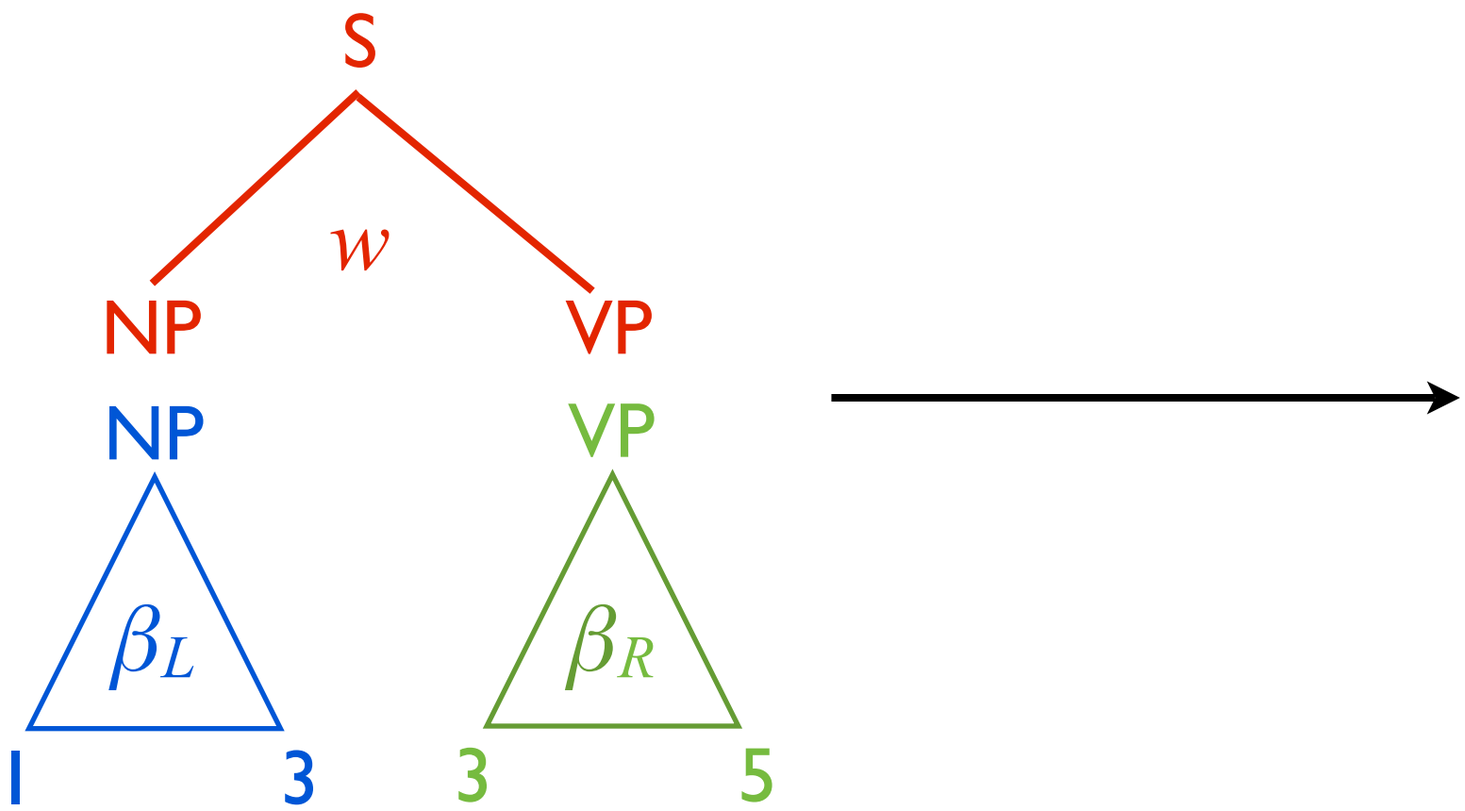
Building Edges



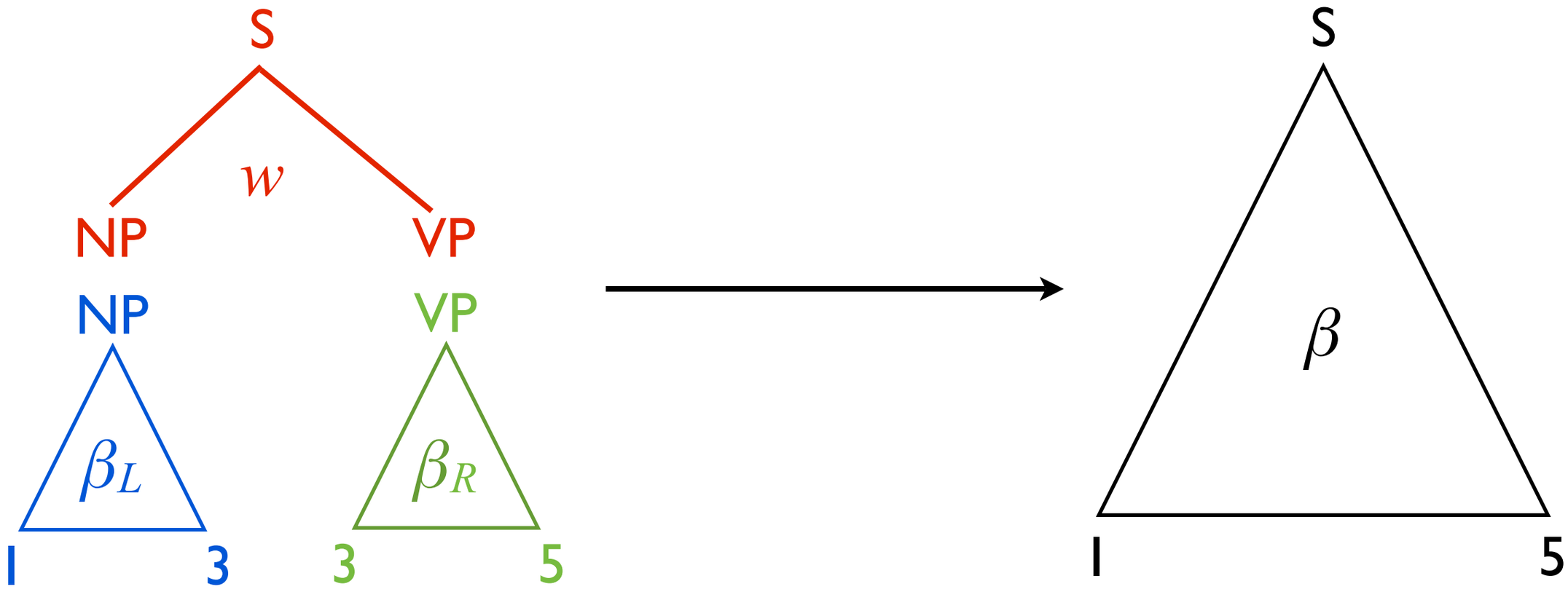
Building Edges



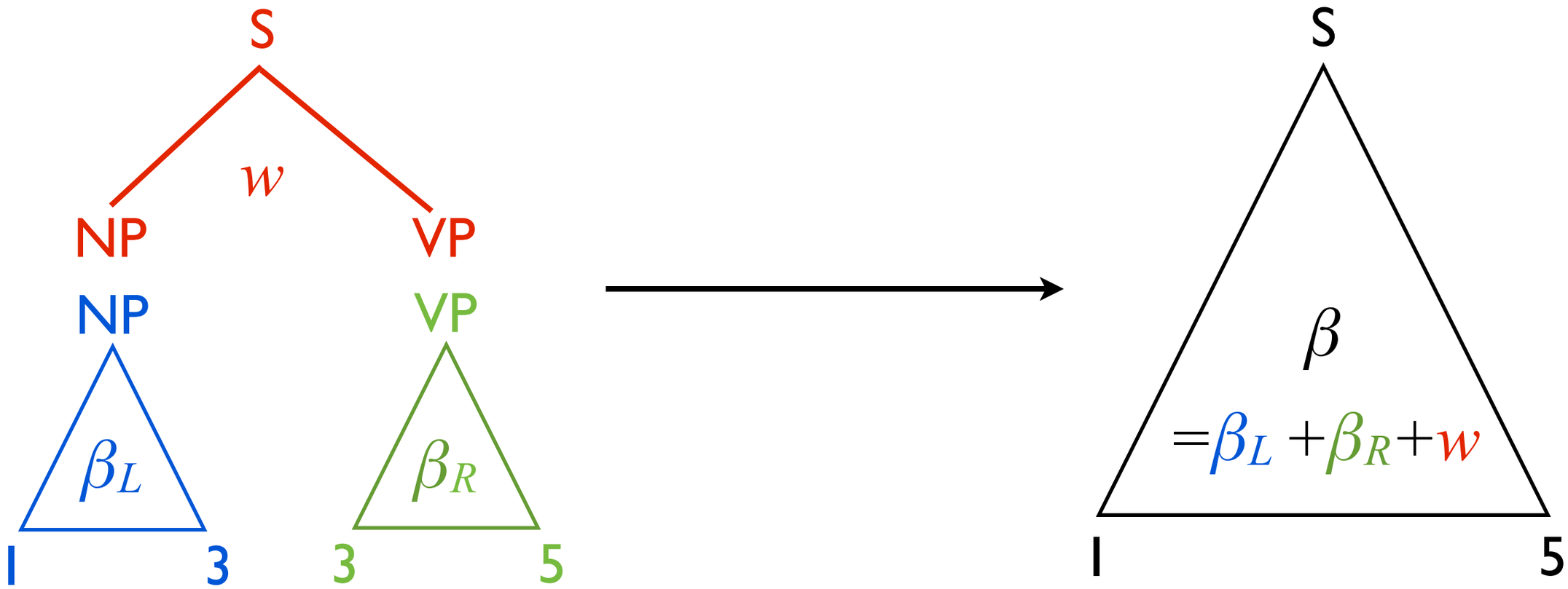
Building Edges



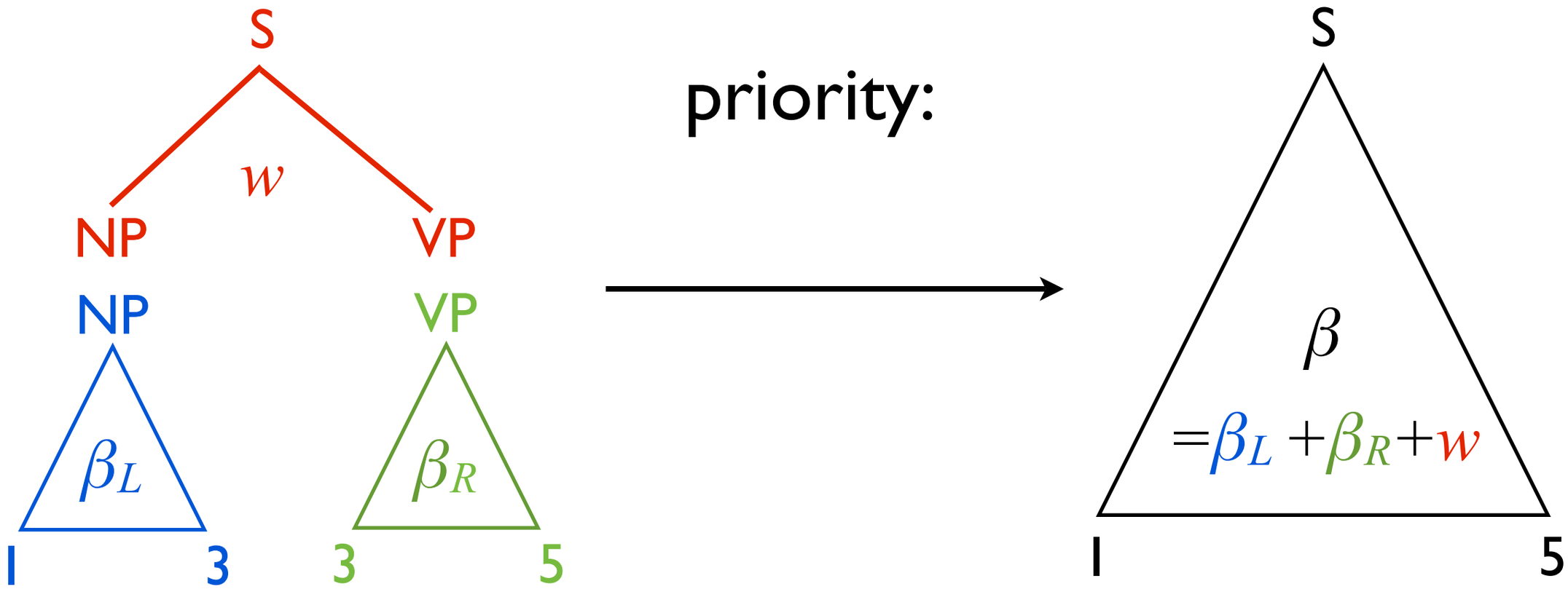
Building Edges



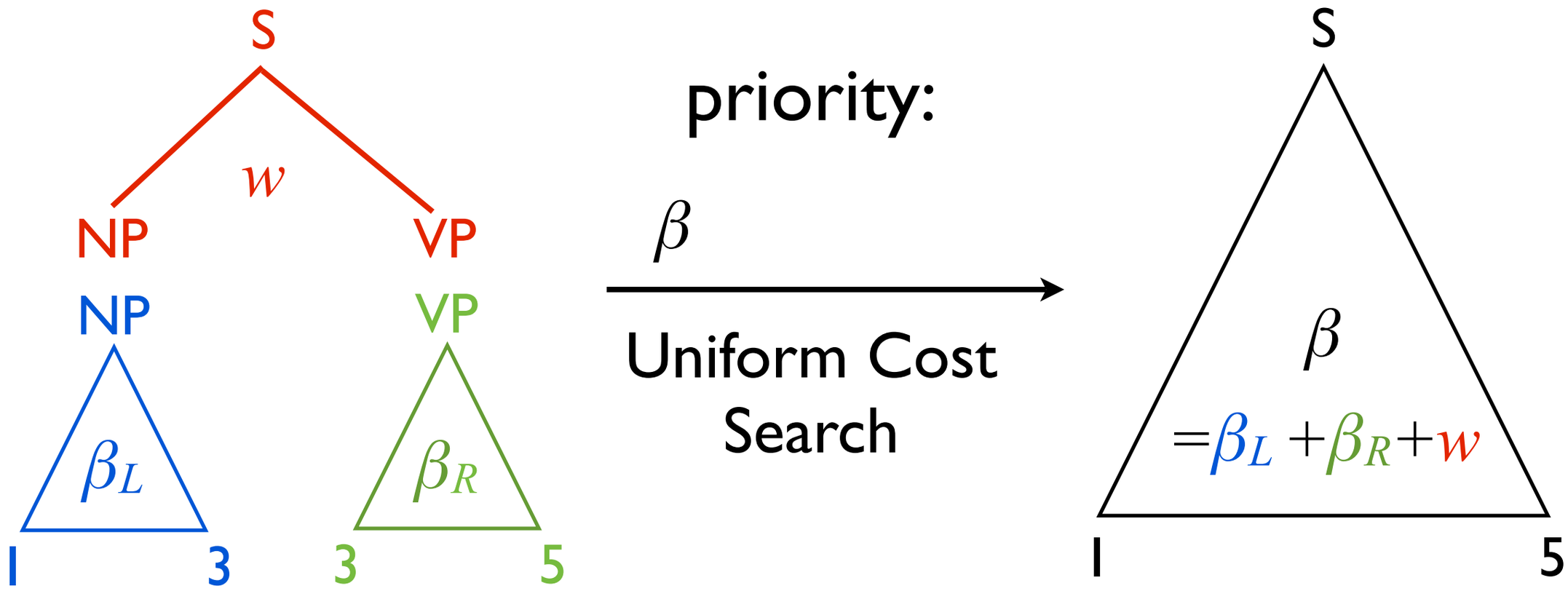
Building Edges



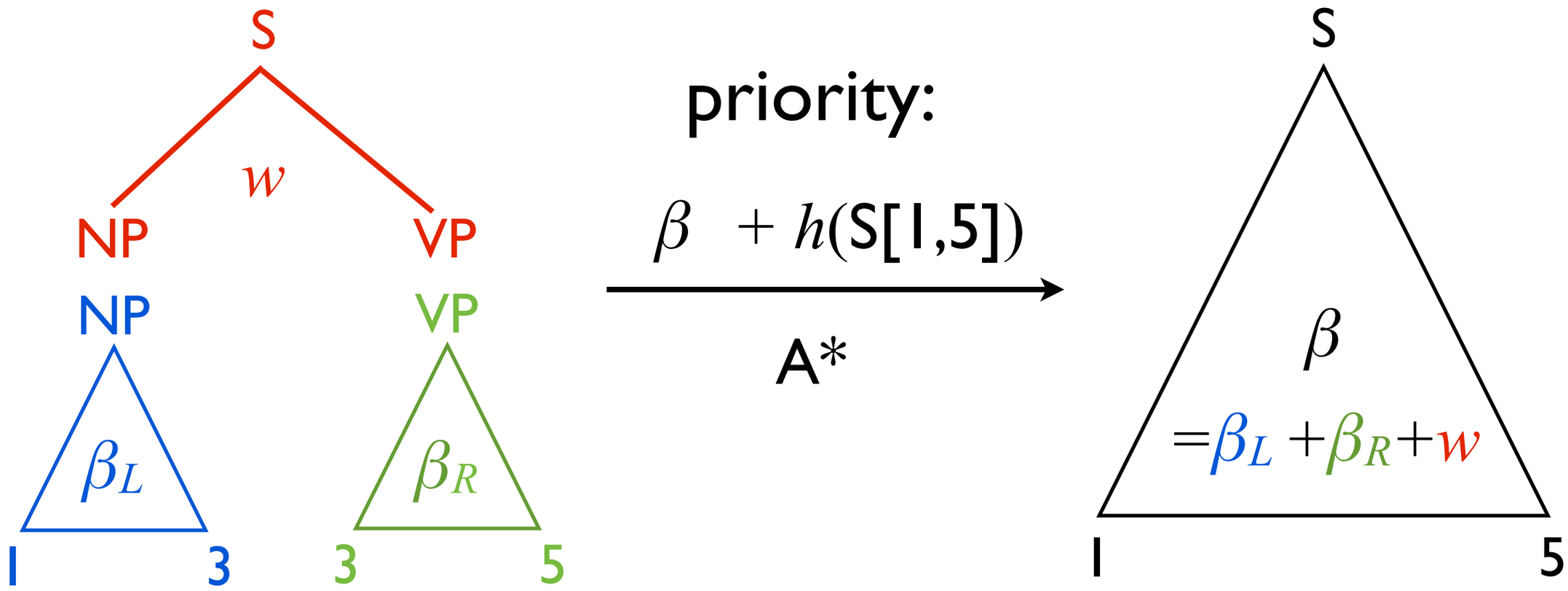
Building Edges



Building Edges



Building Edges



Heuristics

- h is a *heuristic* which lower bounds the Viterbi outside cost α

Heuristics

- h is a *heuristic* which lower bounds the Viterbi outside cost α

$$h(\text{VP}[2,4]) \leq \alpha(\text{VP}[2,4])$$

Heuristics

- h is a *heuristic* which lower bounds the Viterbi outside cost α

$$h(\text{VP}[2,4]) \leq \alpha(\text{VP}[2,4])$$

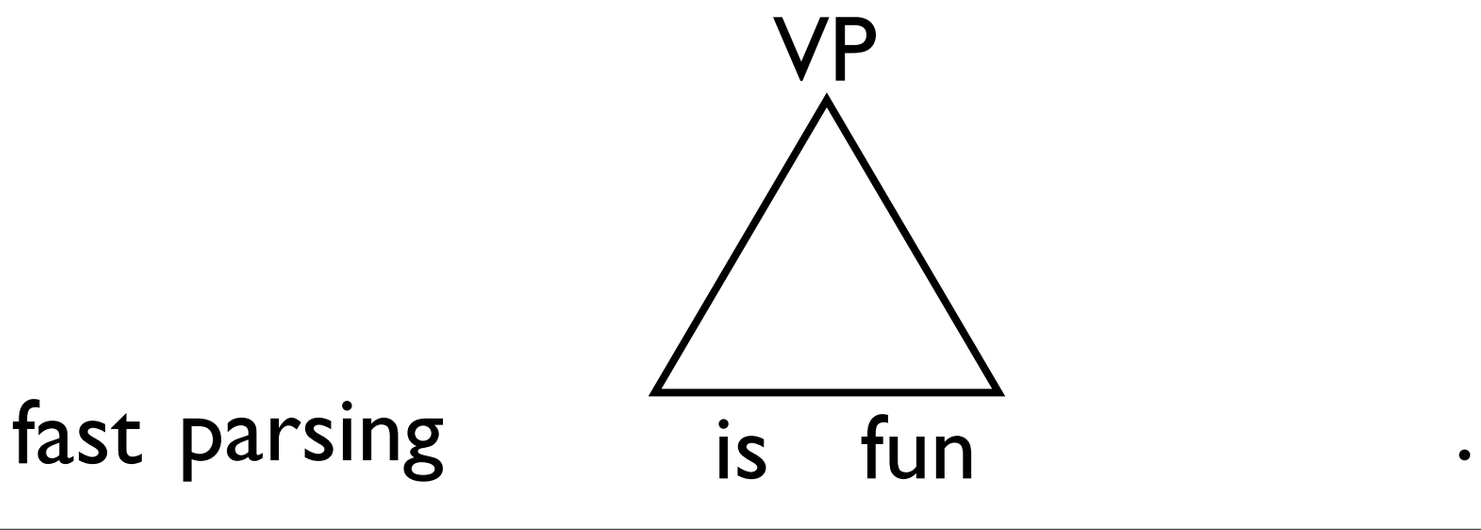
VP

fast parsing is fun .

Heuristics

- h is a *heuristic* which lower bounds the Viterbi outside cost α

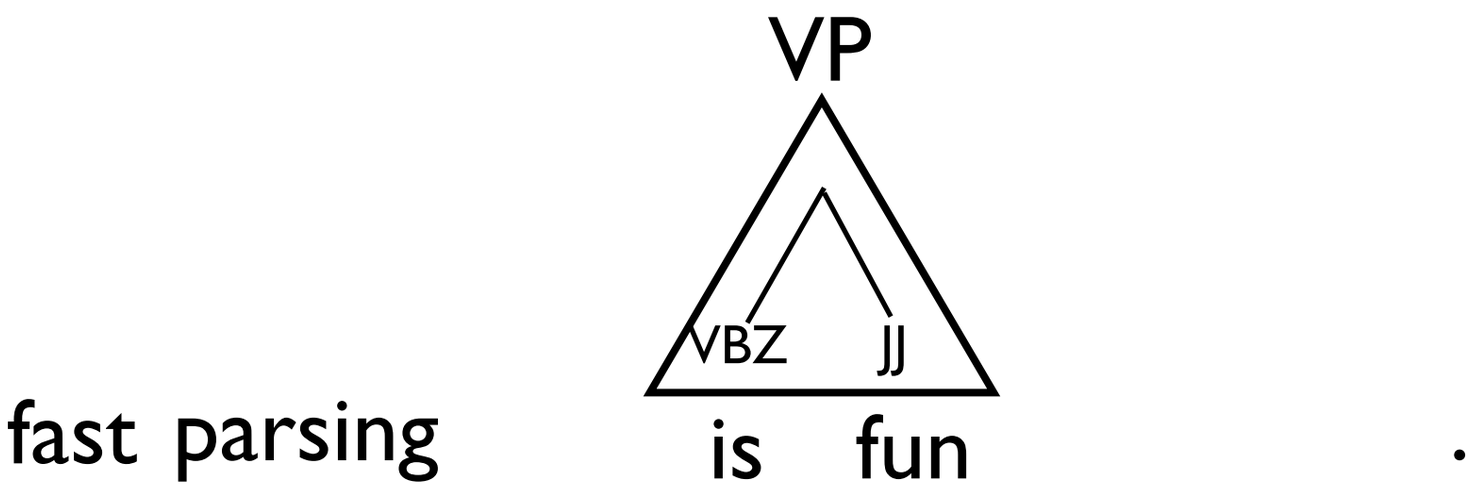
$$h(\text{VP}[2,4]) \leq \alpha(\text{VP}[2,4])$$



Heuristics

- h is a *heuristic* which lower bounds the Viterbi outside cost α

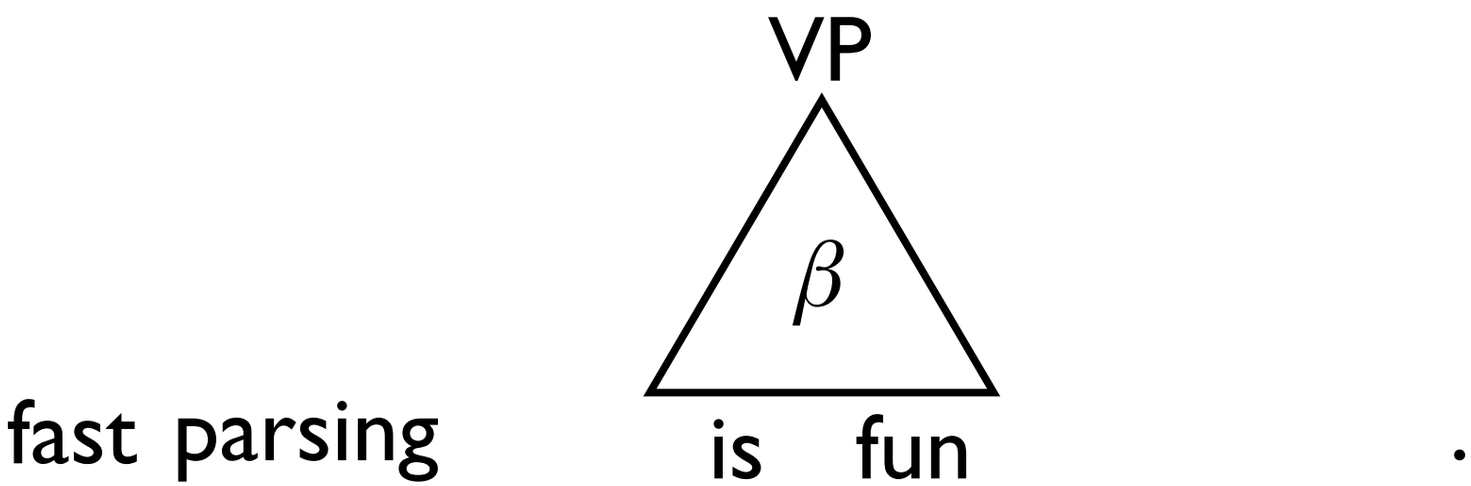
$$h(\text{VP}[2,4]) \leq \alpha(\text{VP}[2,4])$$



Heuristics

- h is a *heuristic* which lower bounds the Viterbi outside cost α

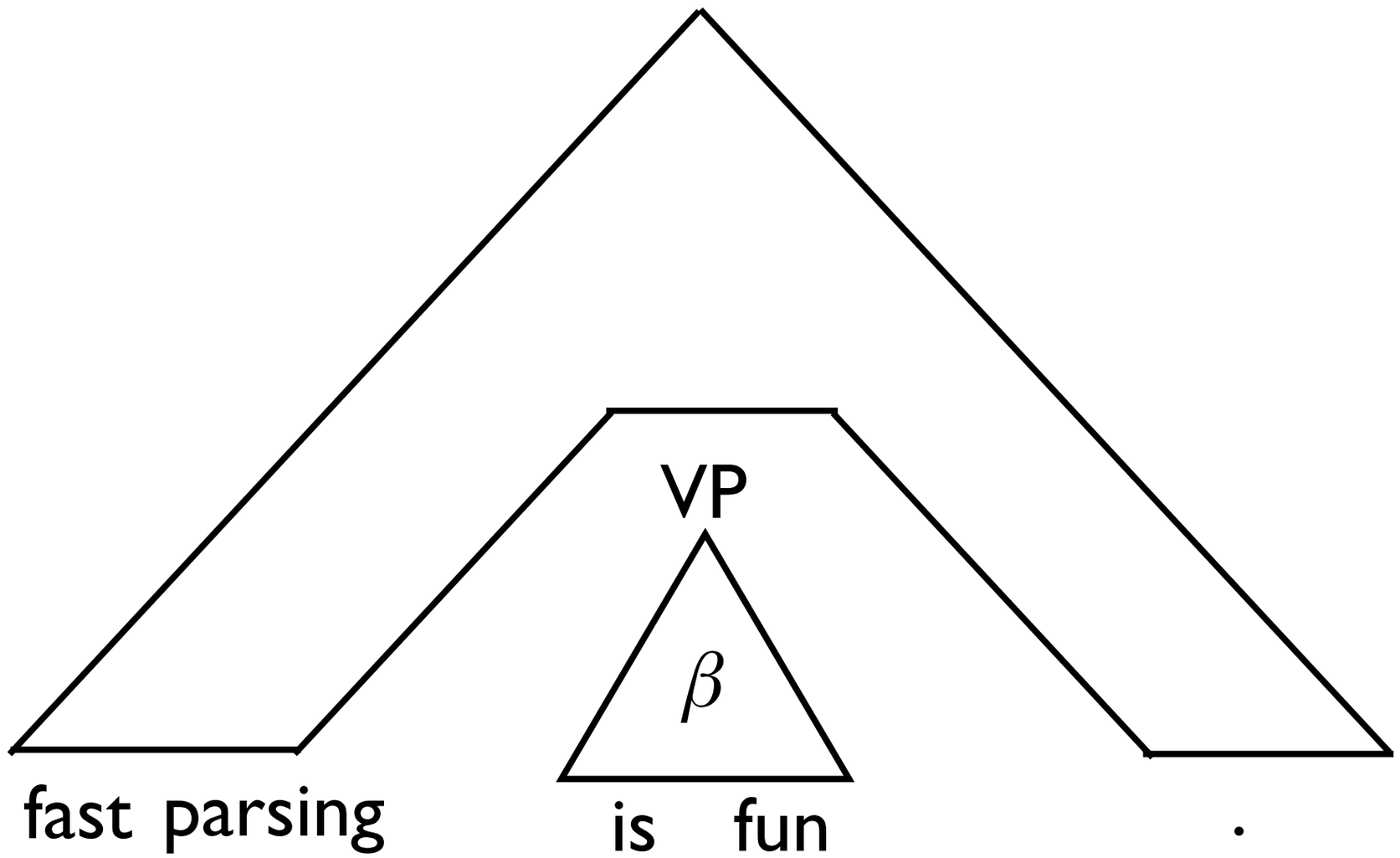
$$h(\text{VP}[2,4]) \leq \alpha(\text{VP}[2,4])$$



Heuristics

- h is a *heuristic* which lower bounds the Viterbi outside cost α

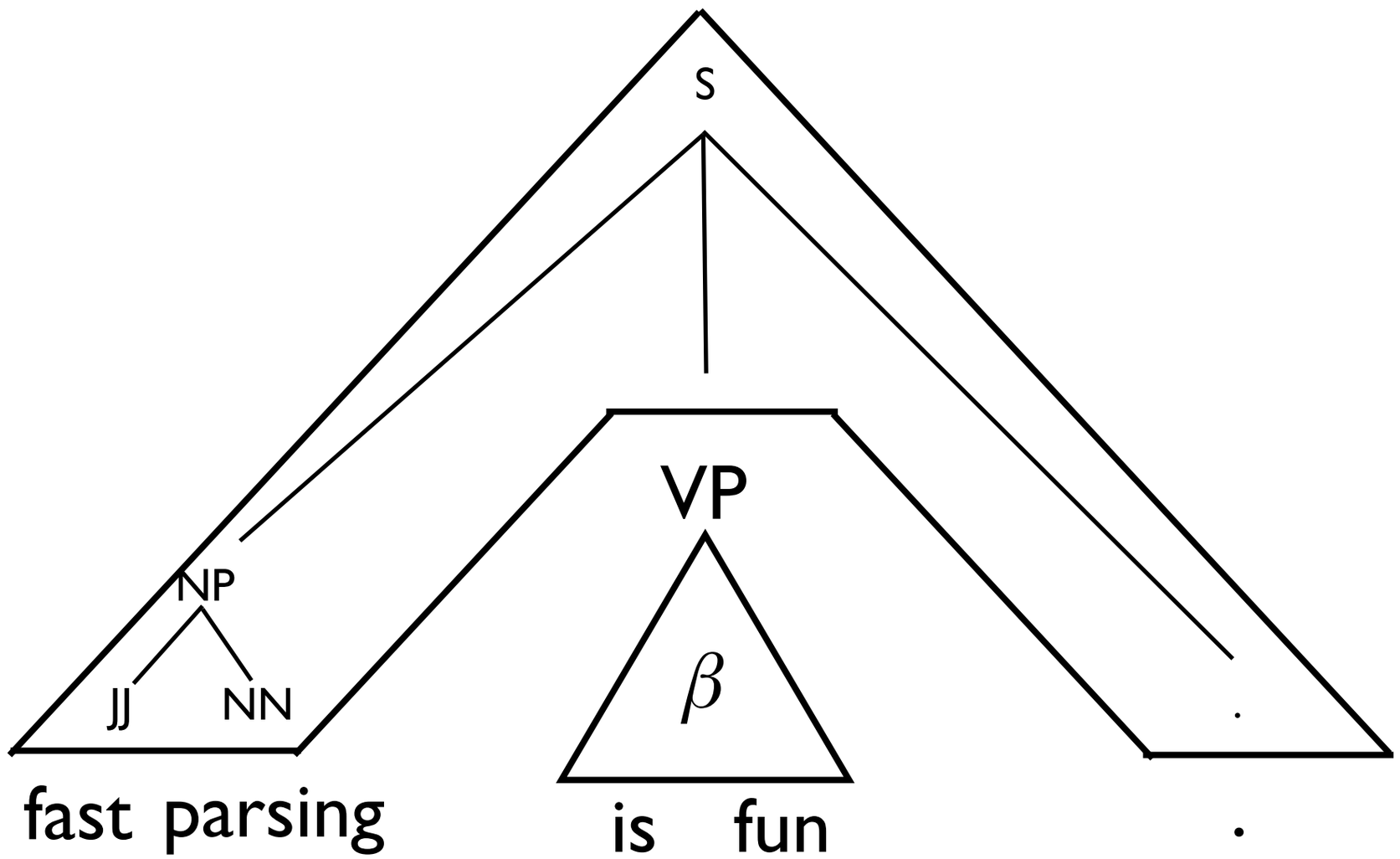
$$h(\text{VP}[2,4]) \leq \alpha(\text{VP}[2,4])$$



Heuristics

- h is a *heuristic* which lower bounds the Viterbi outside cost α

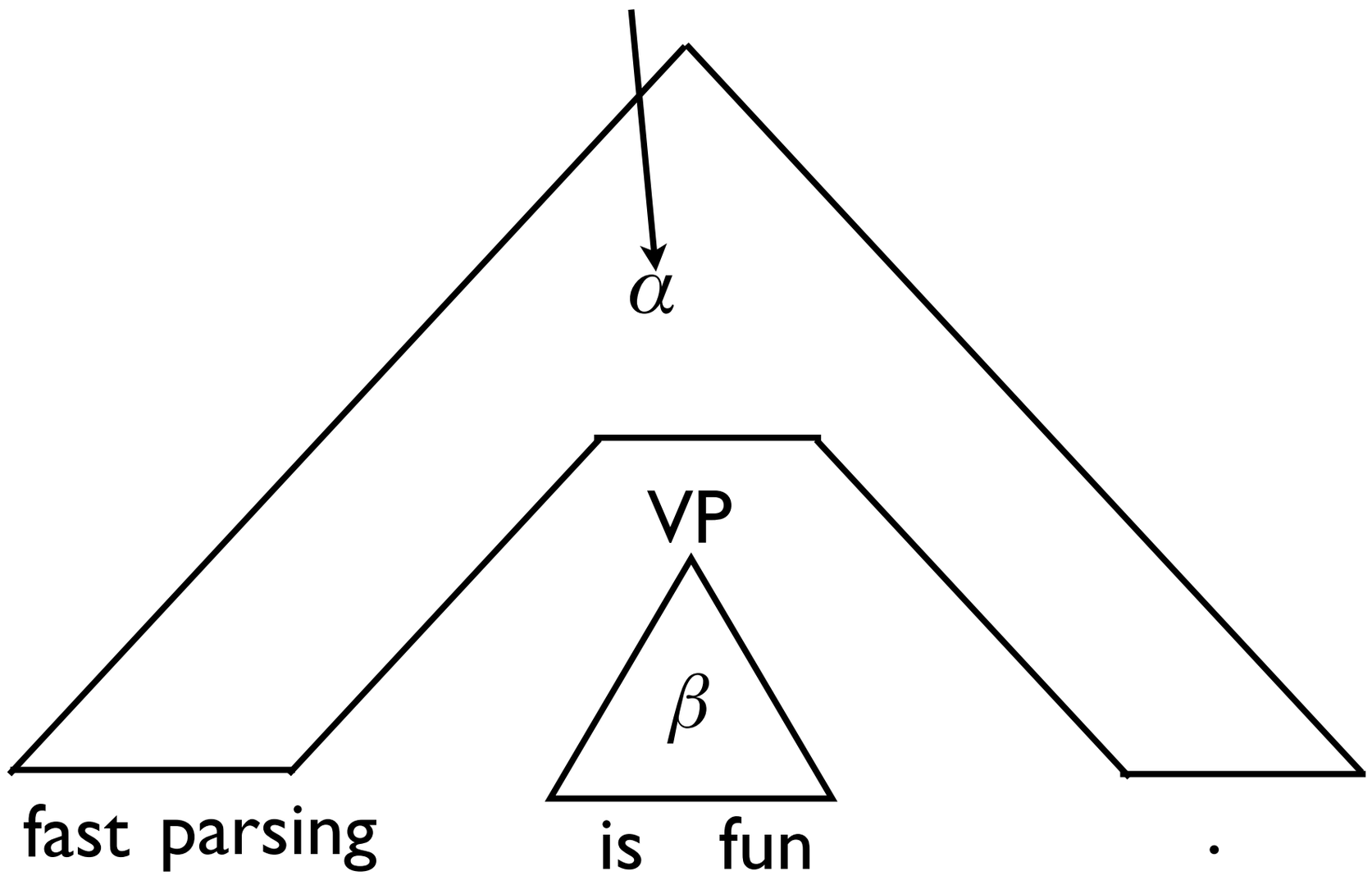
$$h(\text{VP}[2,4]) \leq \alpha(\text{VP}[2,4])$$



Heuristics

- h is a *heuristic* which lower bounds the Viterbi outside cost α

$$h(\text{VP}[2,4]) \leq \alpha(\text{VP}[2,4])$$

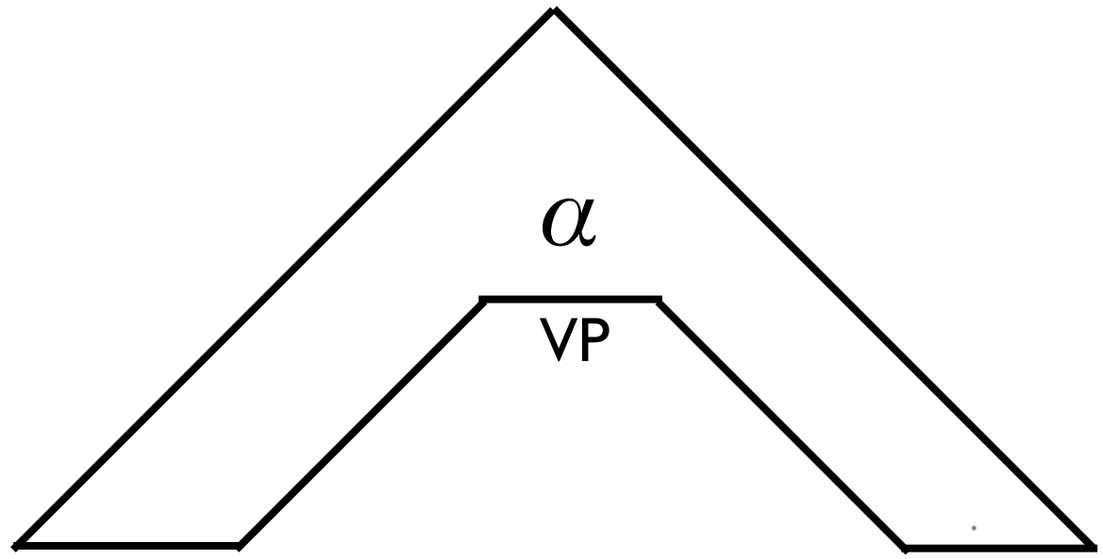


Outside Scores

- We can get lower bounds on α from coarse grammars

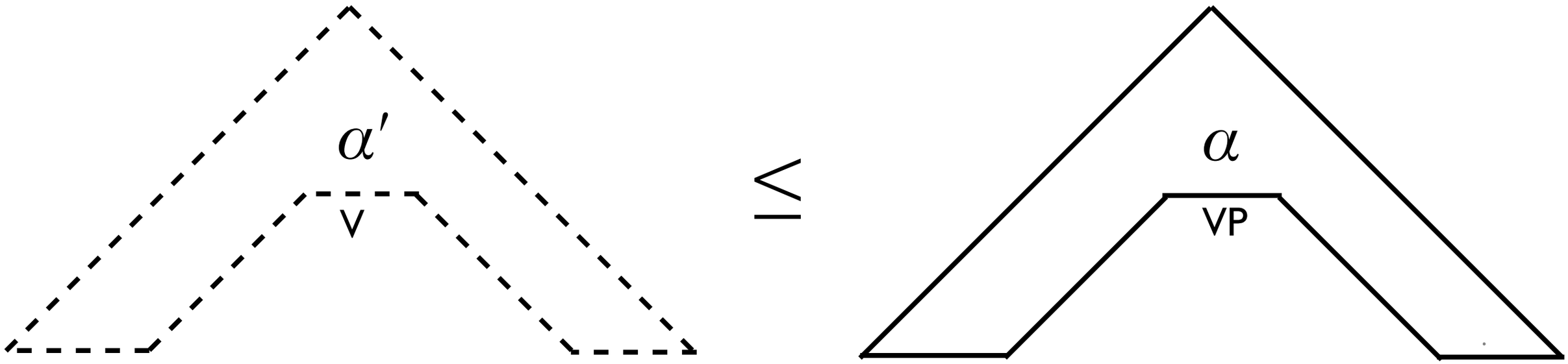
Outside Scores

- We can get lower bounds on α from coarse grammars



Outside Scores

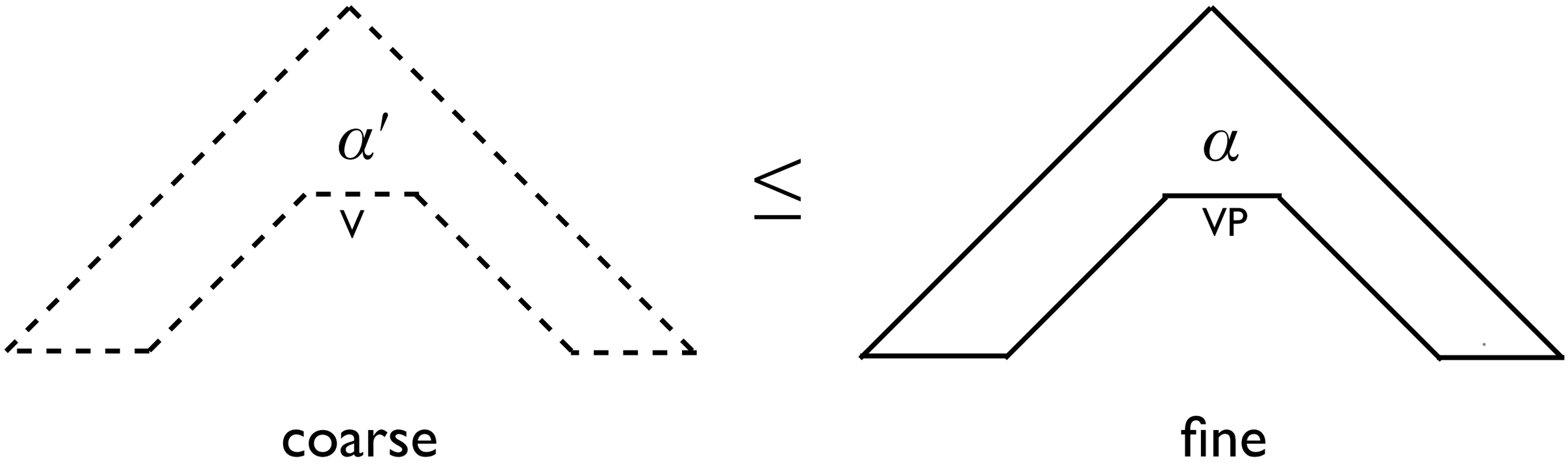
- We can get lower bounds on α from coarse grammars



coarse fine

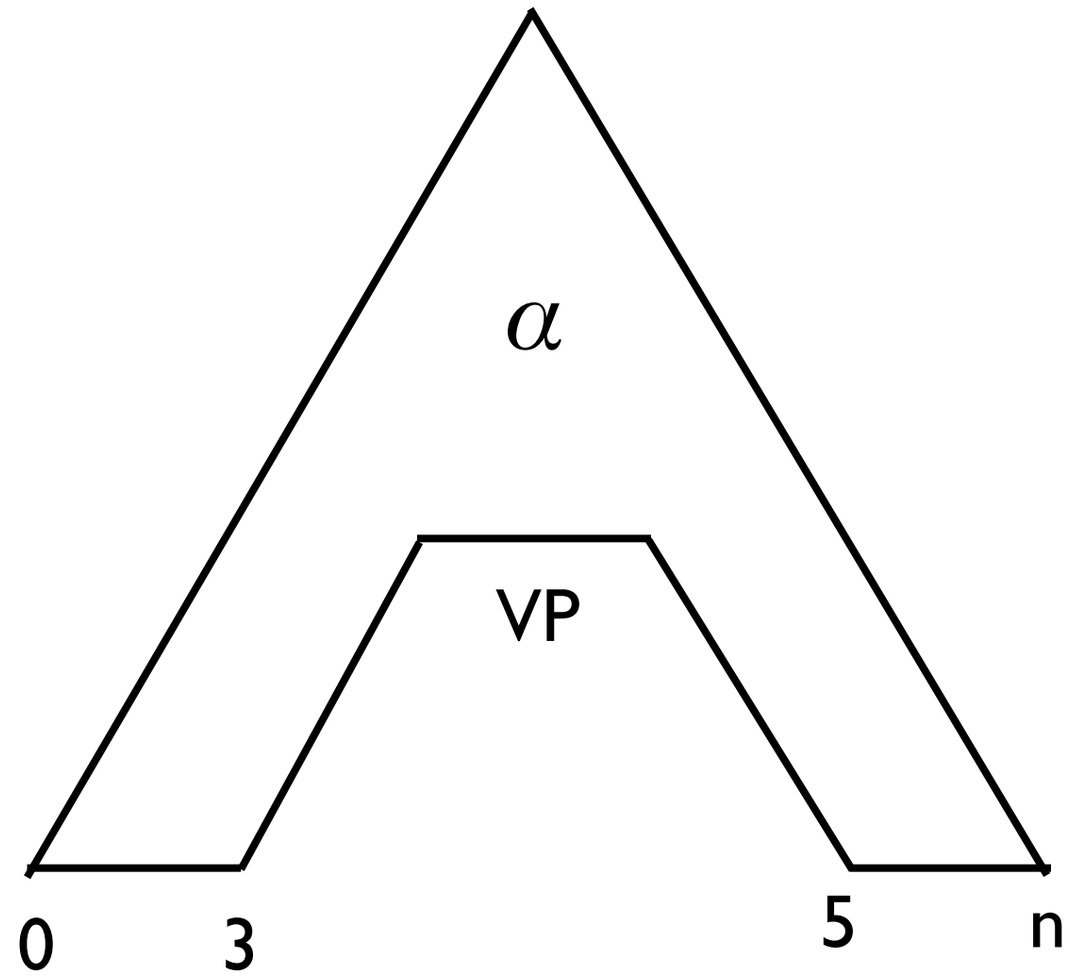
Outside Scores

- We can get lower bounds on α from coarse grammars

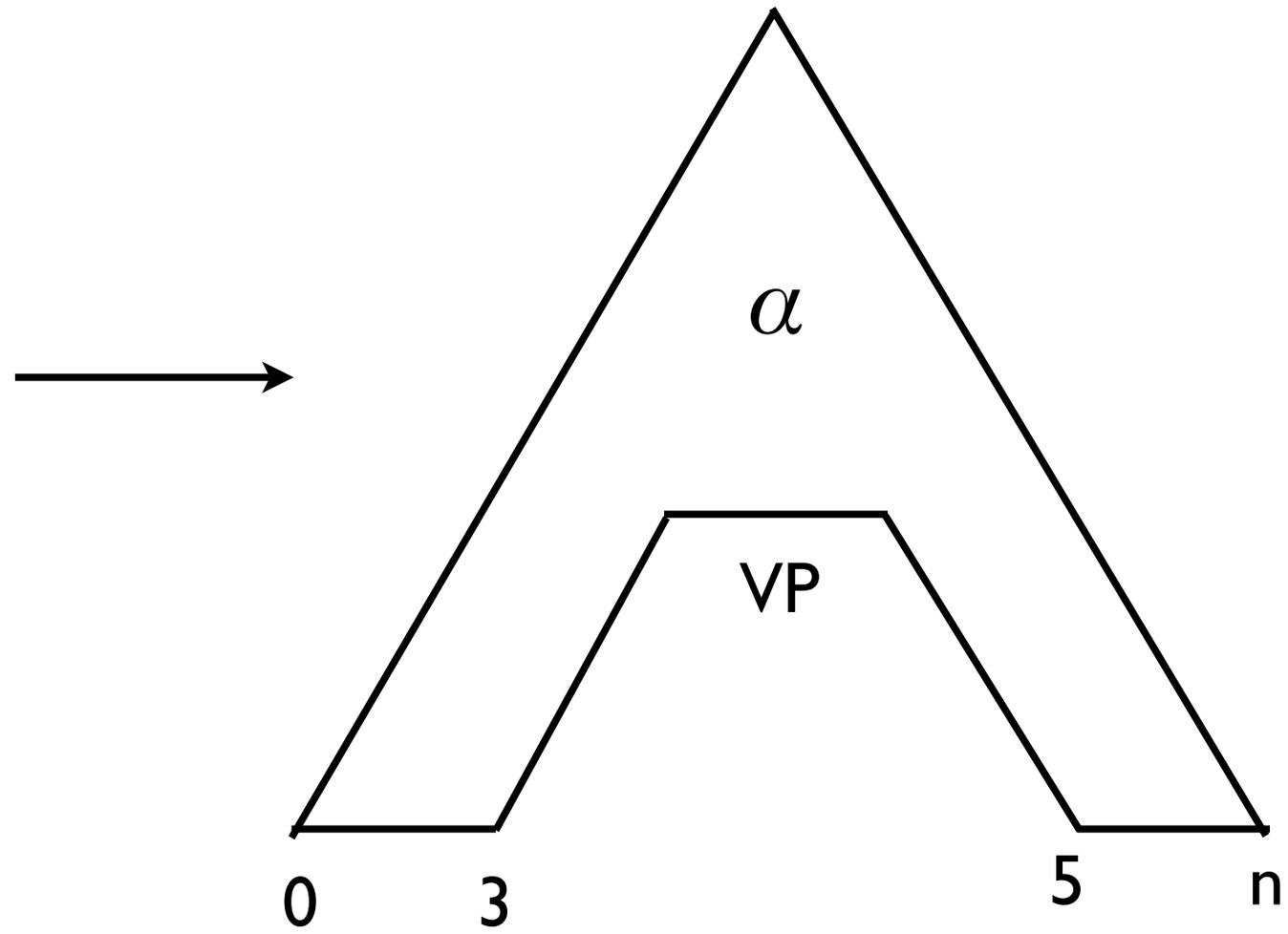


- How do we compute these outside scores?

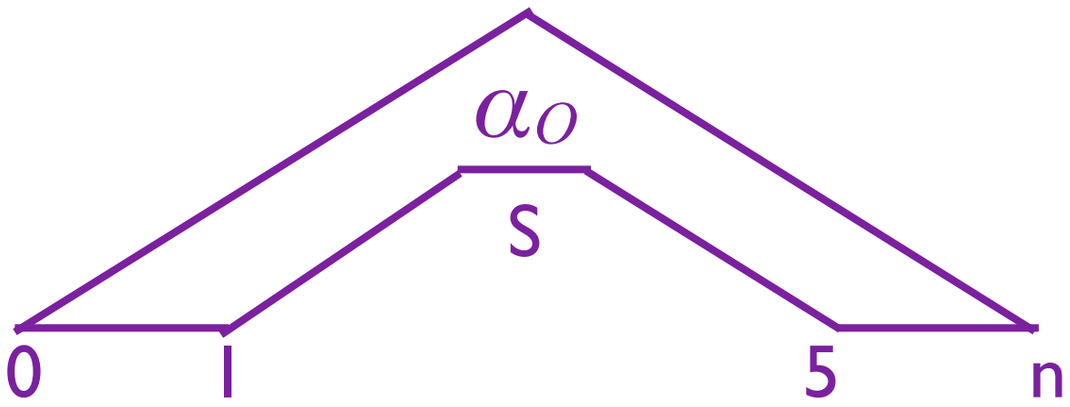
Building Outside Edges

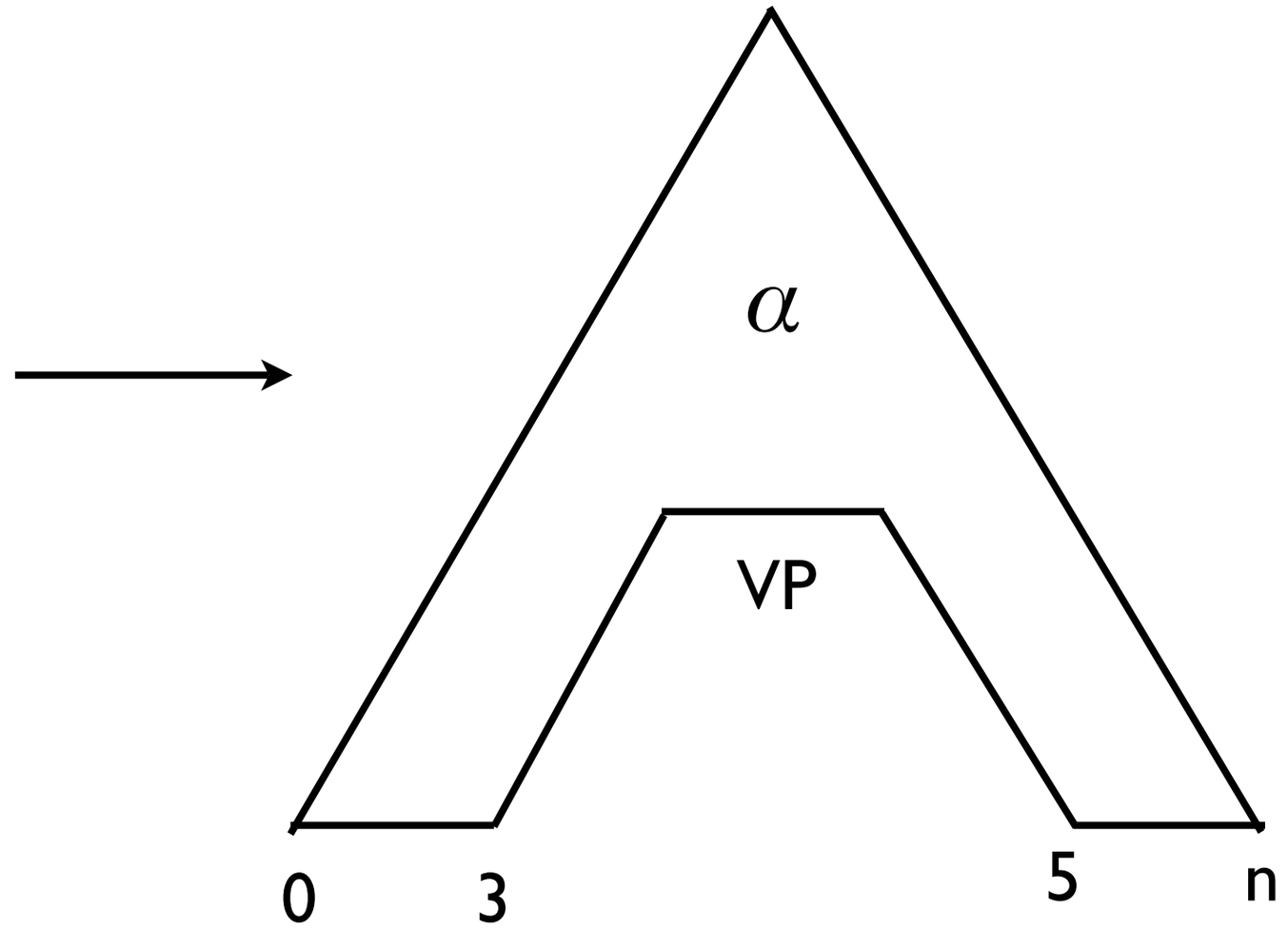


Building Outside Edges

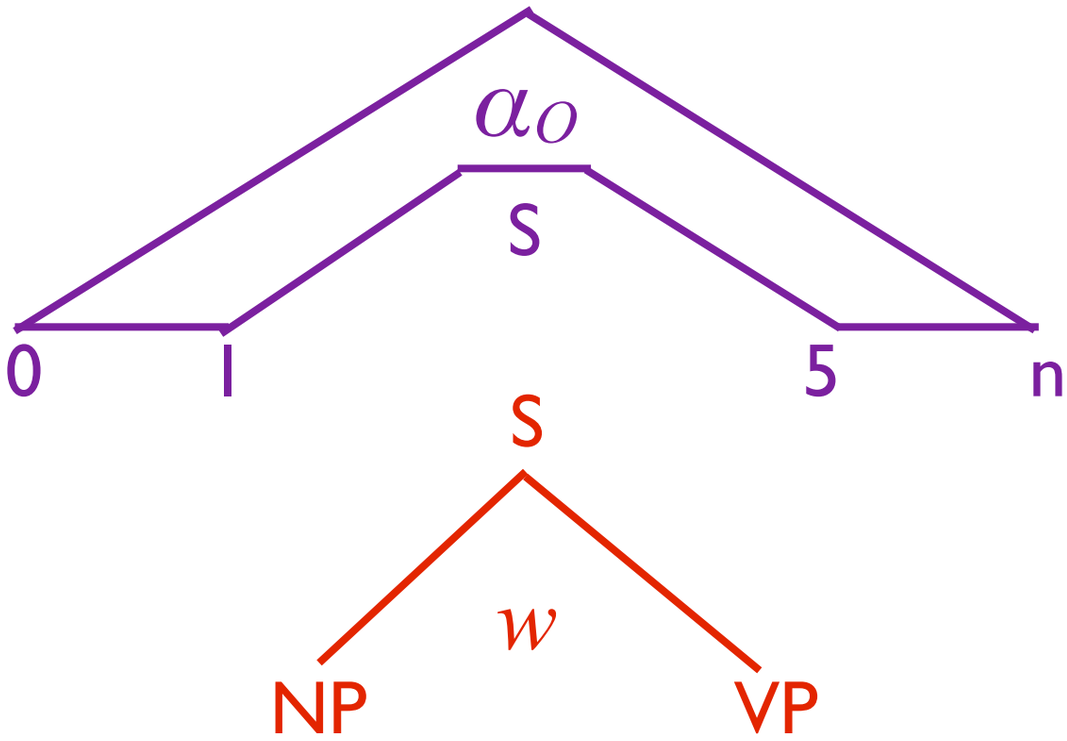


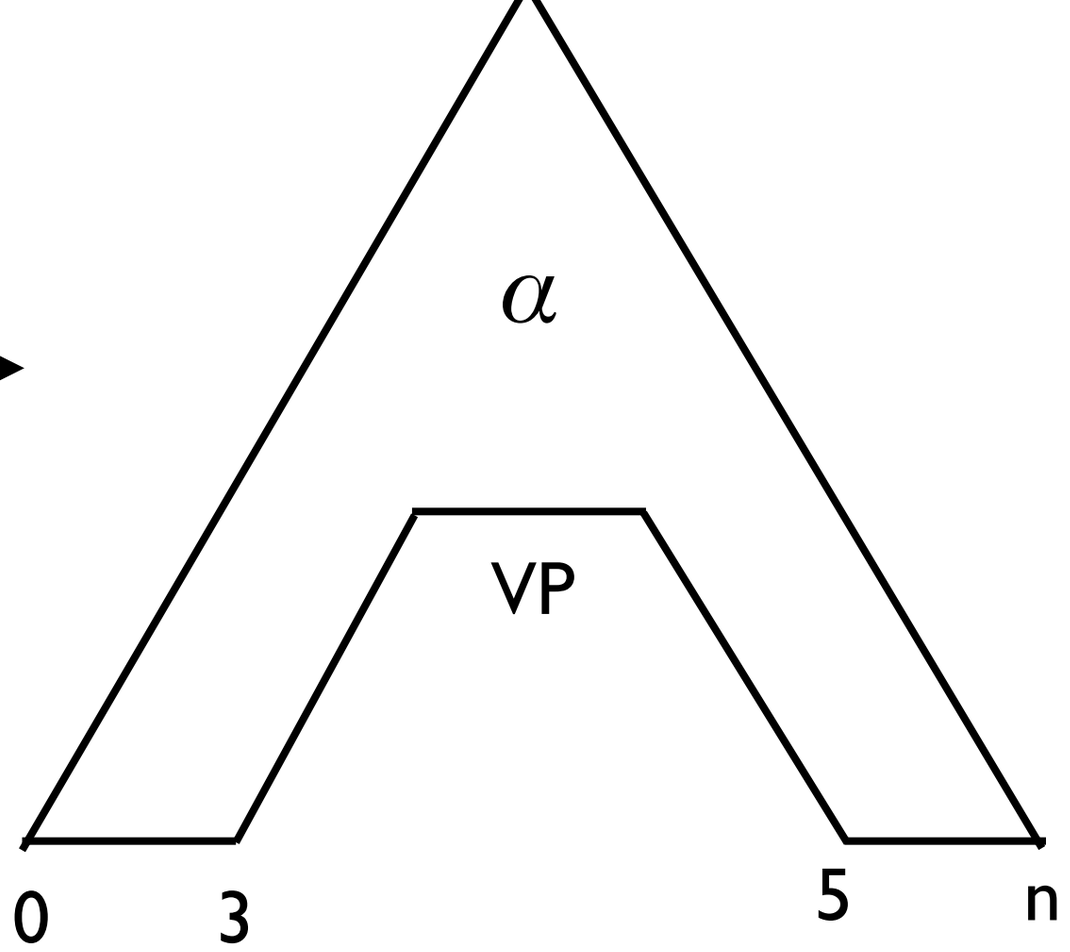
Building Outside Edges



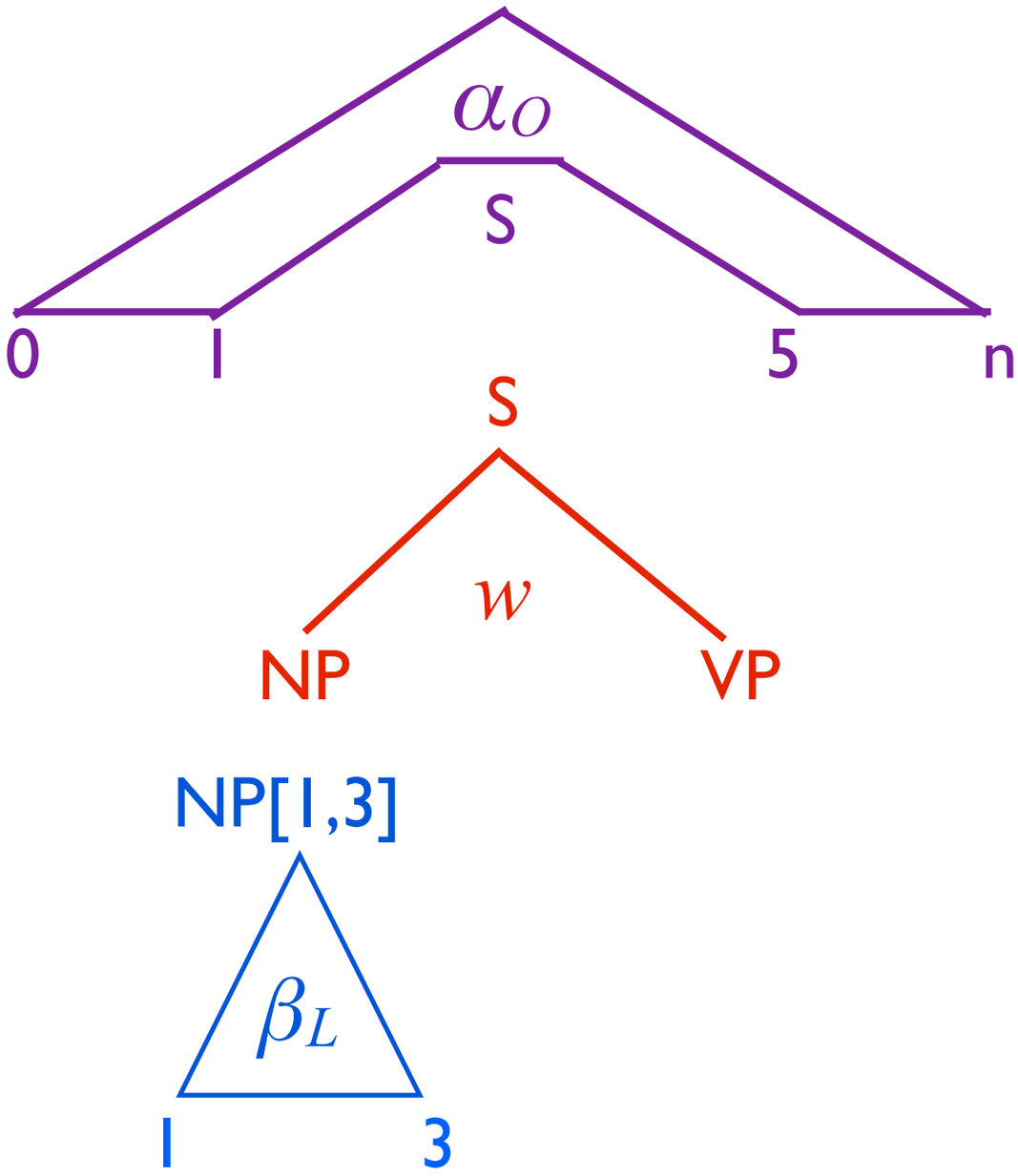


Building Outside Edges

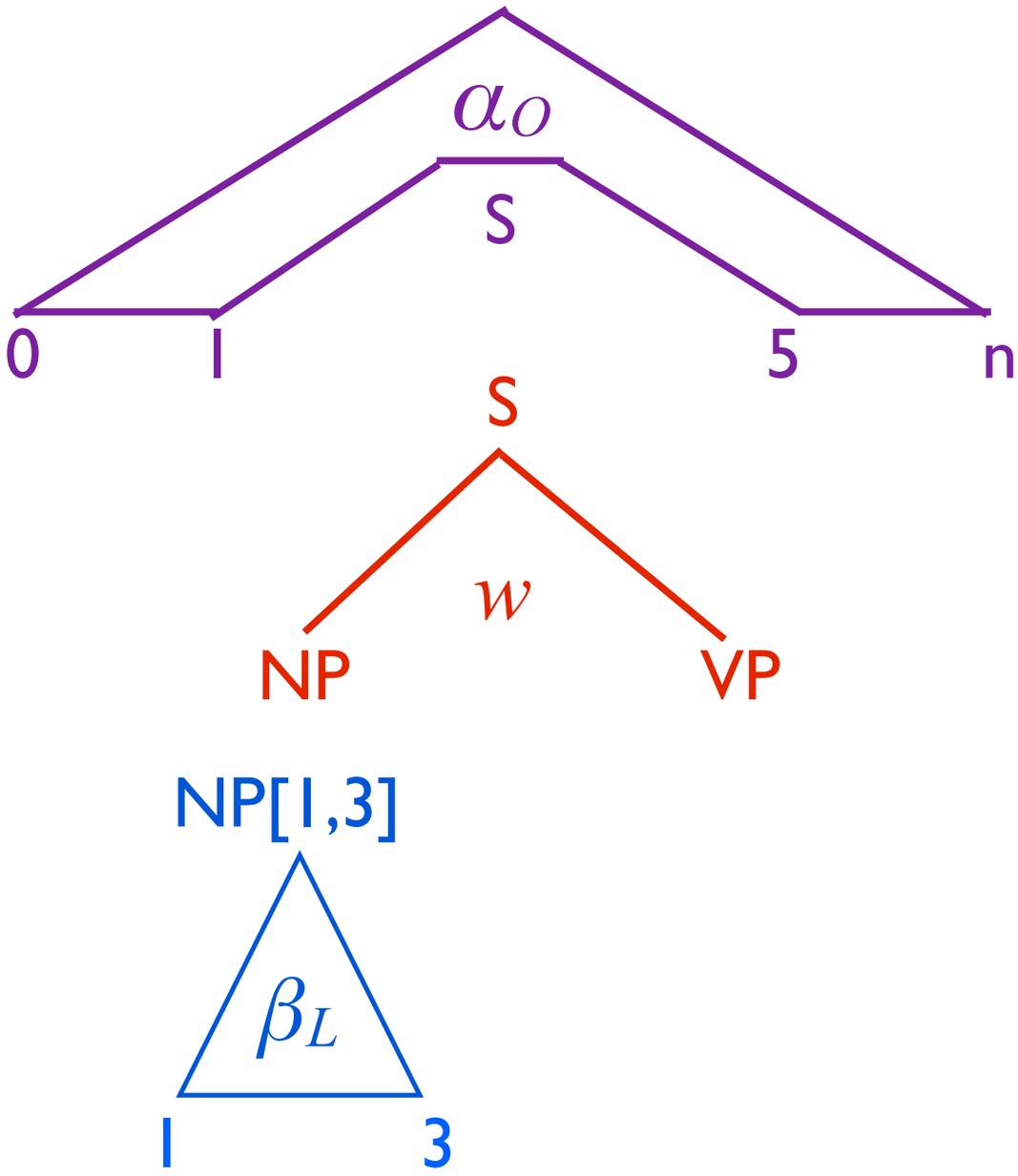


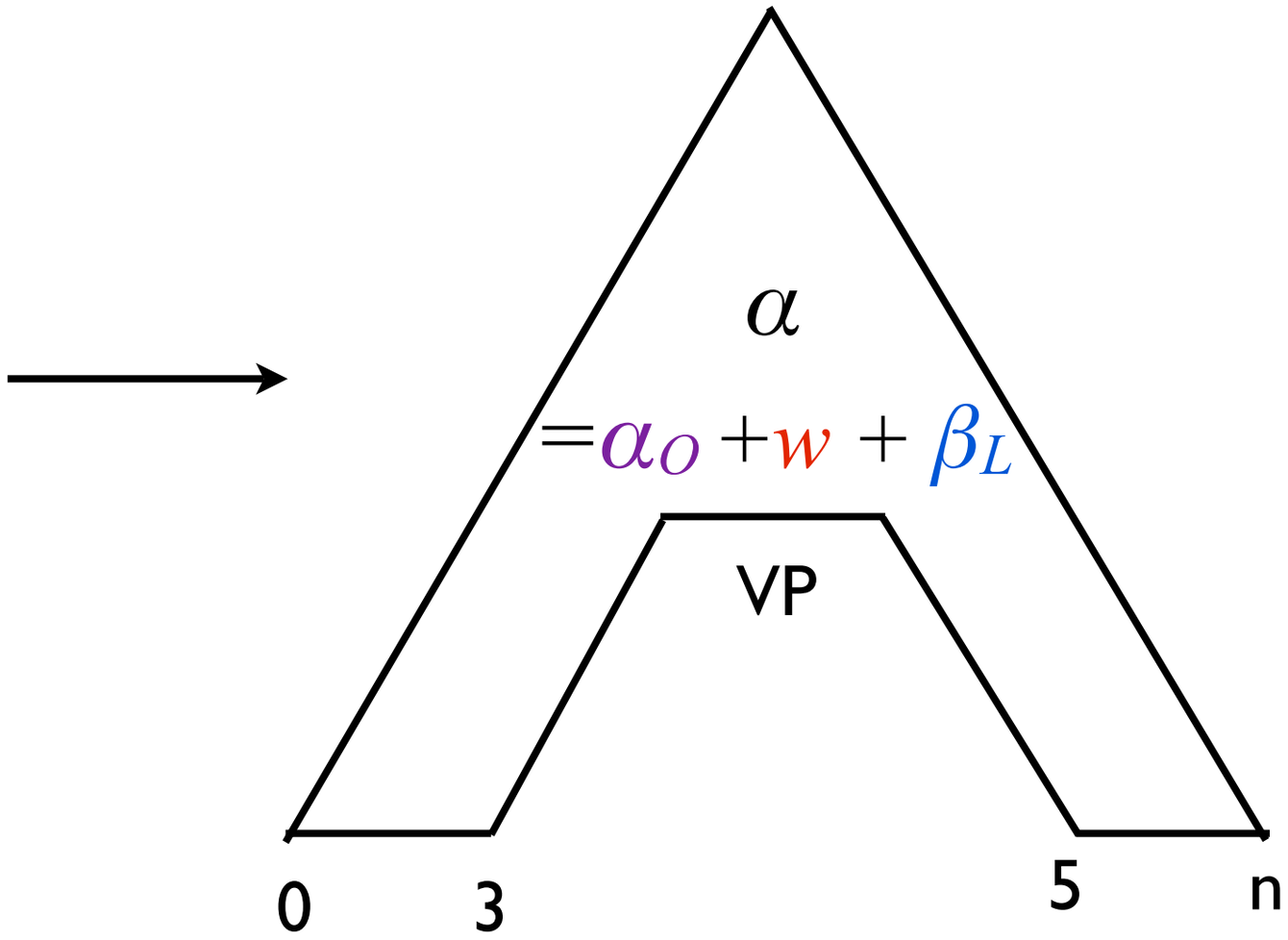


Building Outside Edges



Building Outside Edges

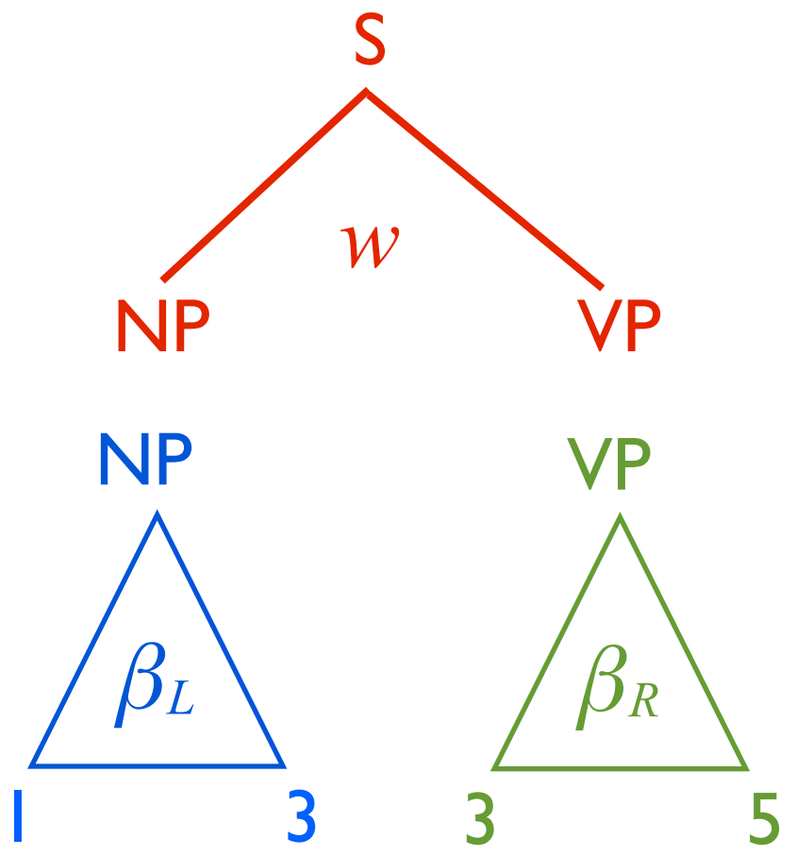




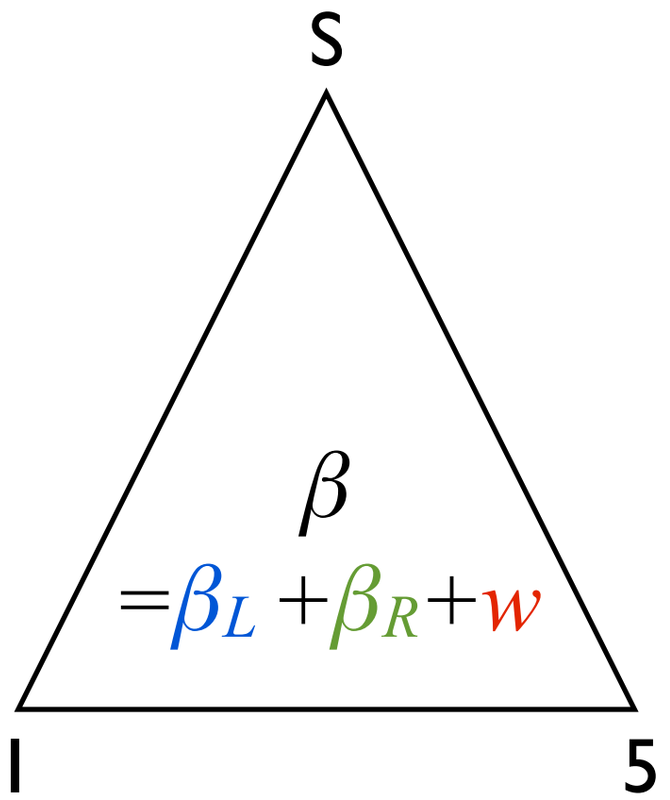
Hierarchical A* (Felzenswalb and McAllester 2007)

- Basic Idea:
  - build both inside and outside edges as needed using same agenda
  - use coarse outside scores as heuristics for fine inside edges

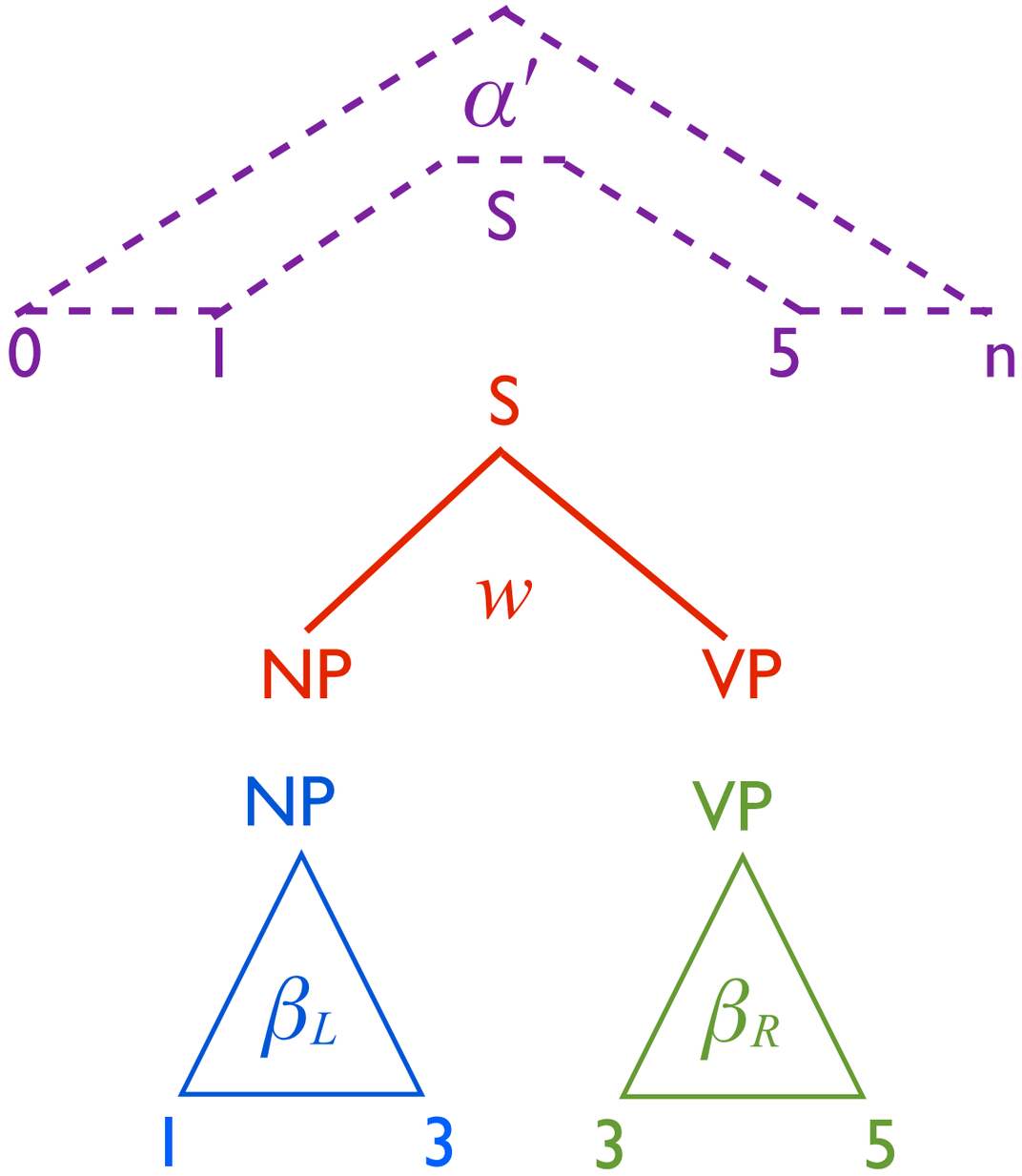
Hierarchically Building Inside Edges



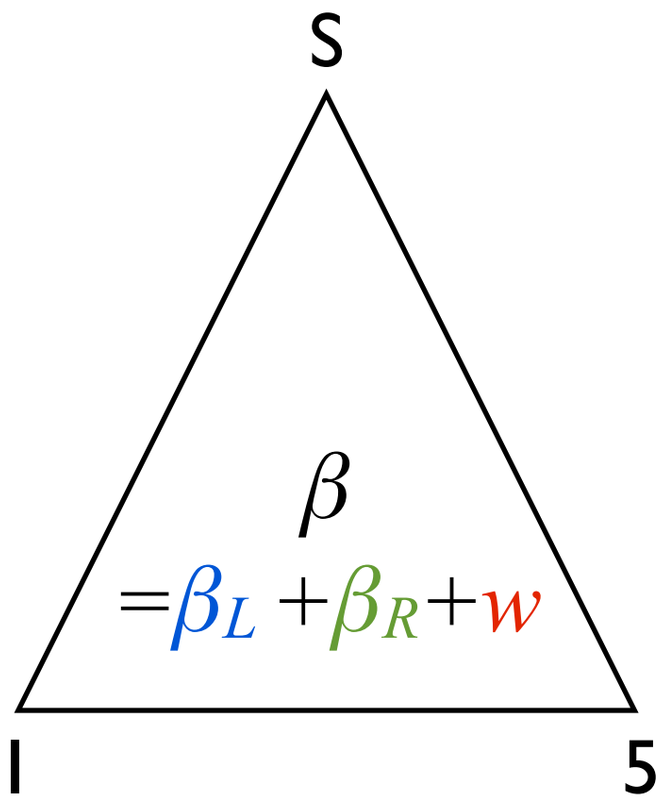
$\beta + h(\text{S}[1,5])$



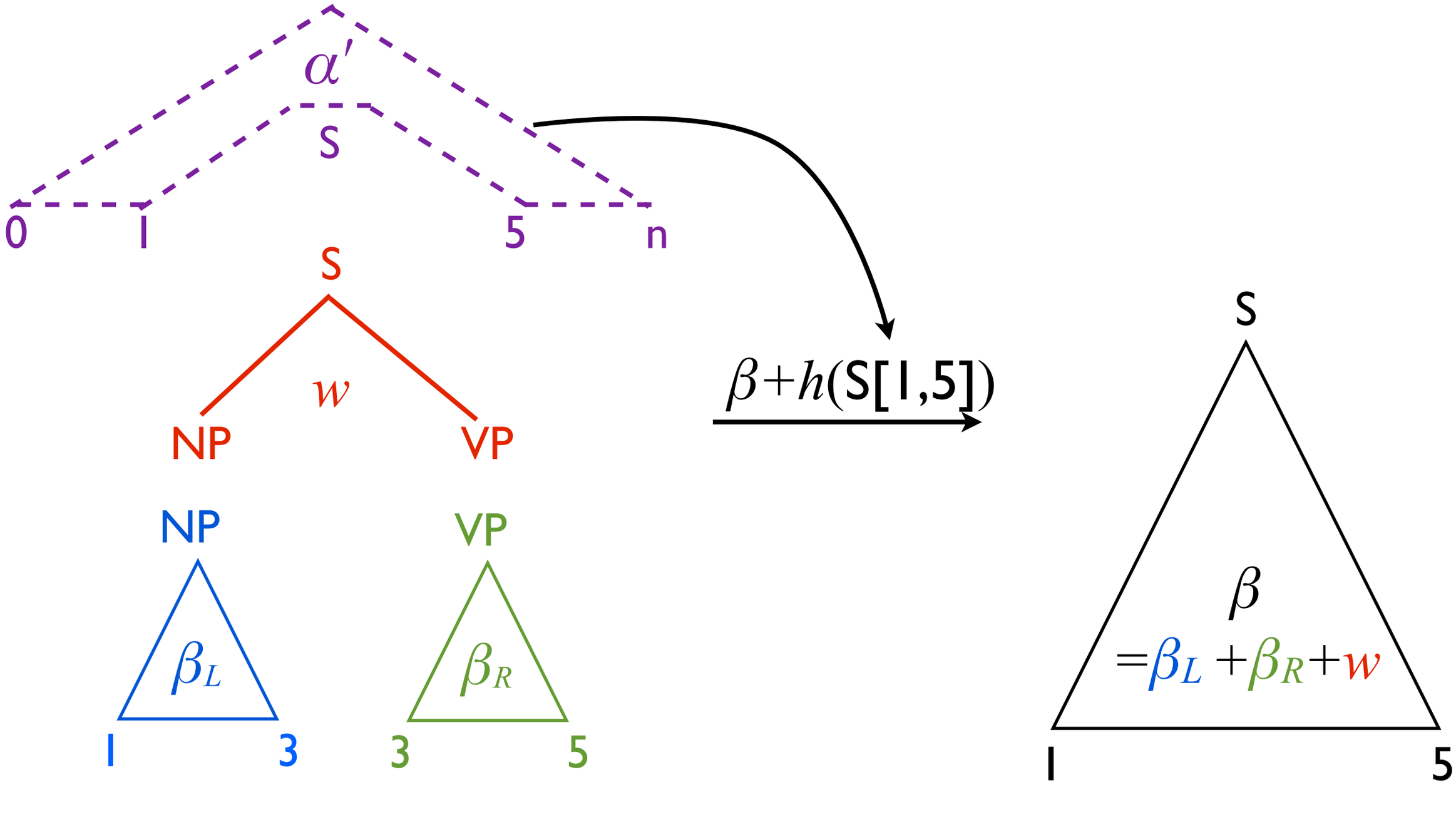
Hierarchically Building Inside Edges



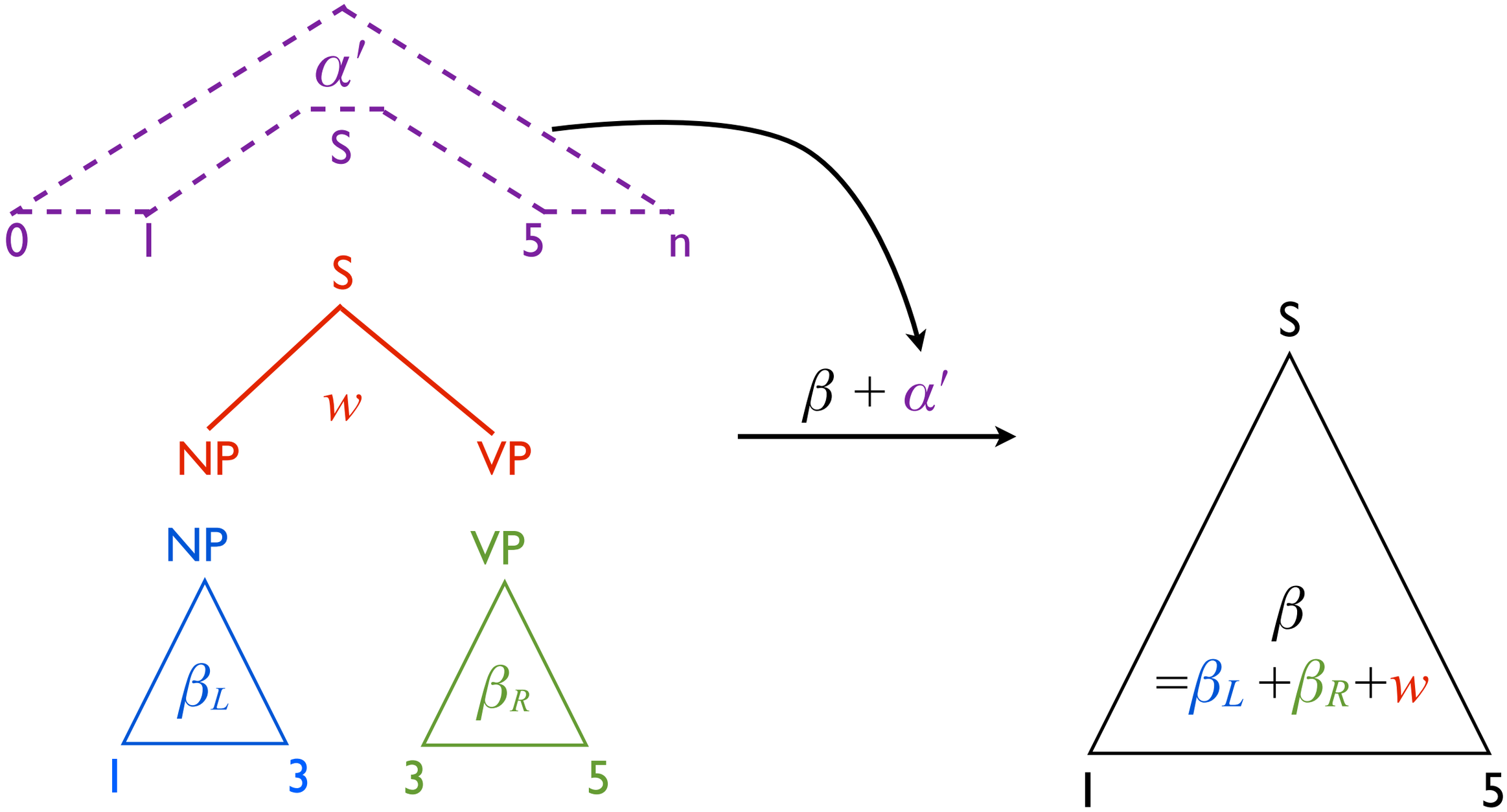
$\beta + h(\mathrm{S}[1,5])$



Hierarchically Building Inside Edges



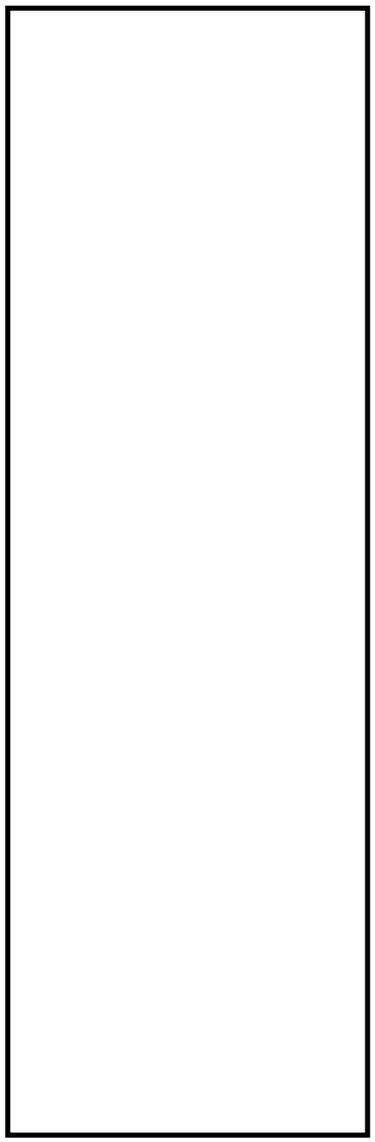
Hierarchically Building Inside Edges



University of California
Berkeley

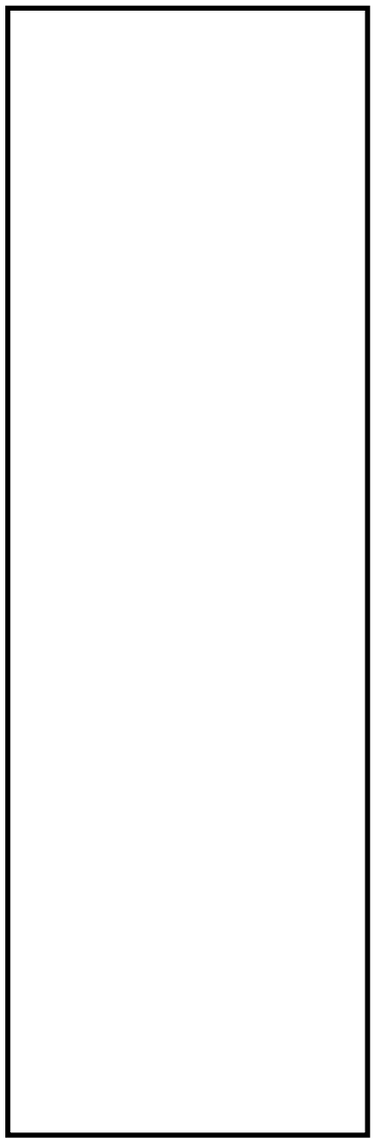
HA*

HA*



Agenda

HA*



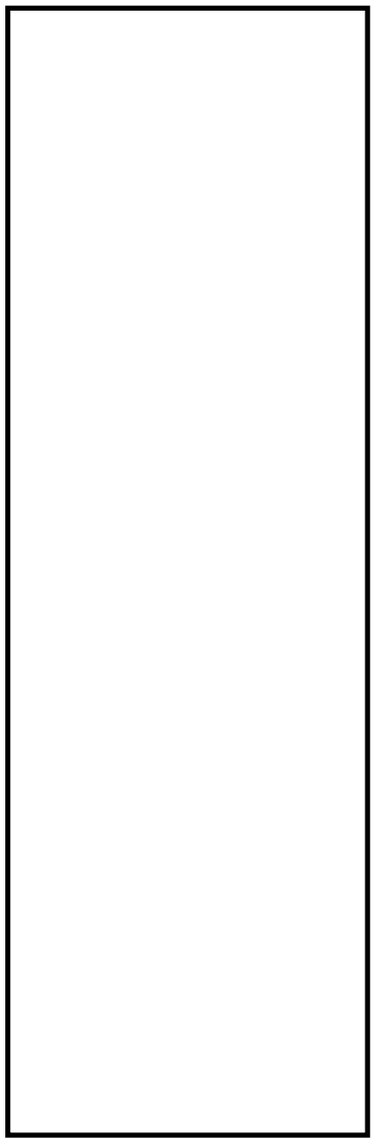
Agenda

Charts

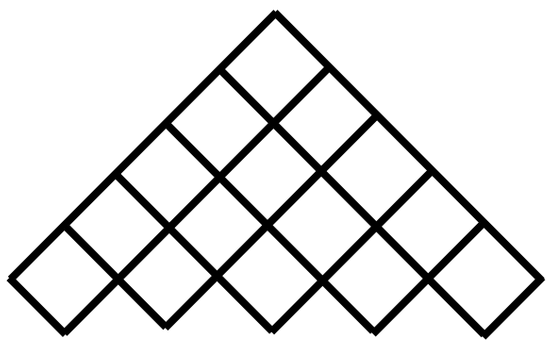
University of California
Berkeley

HA*

inside



Agenda

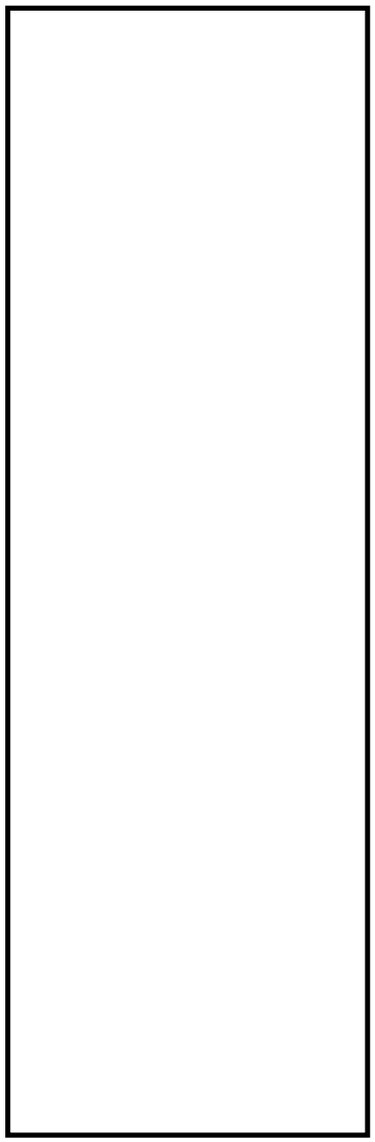


Charts

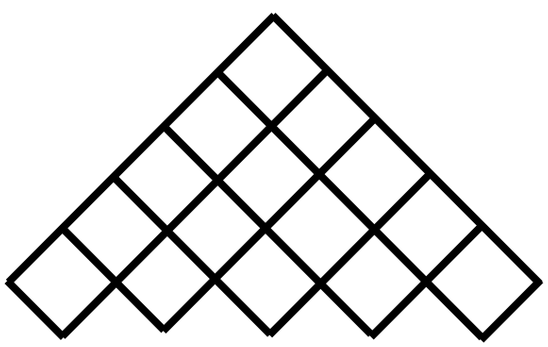
HA*

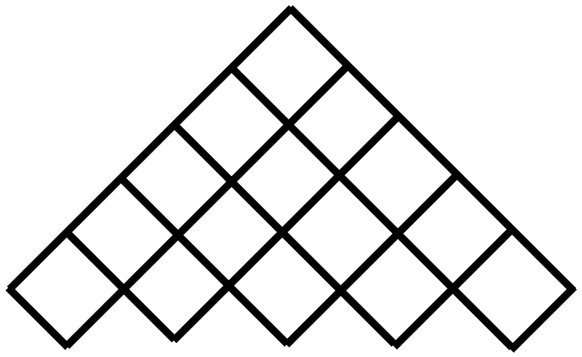
inside

outside



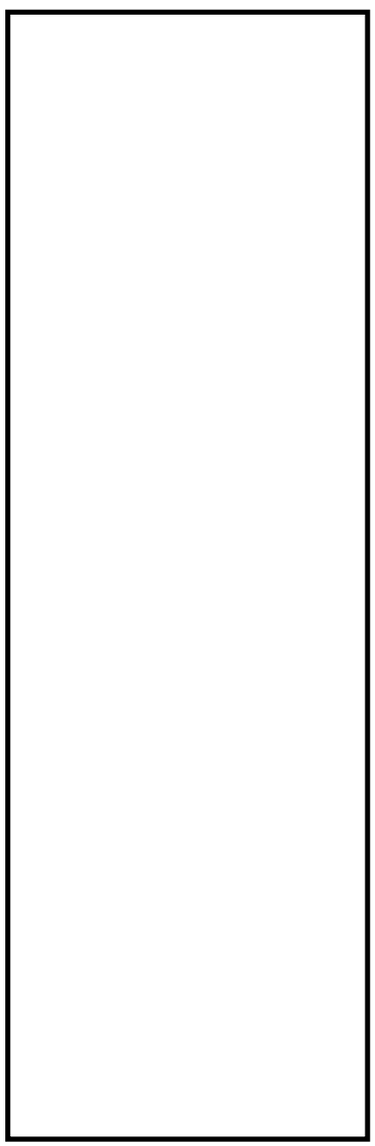
Agenda





Charts

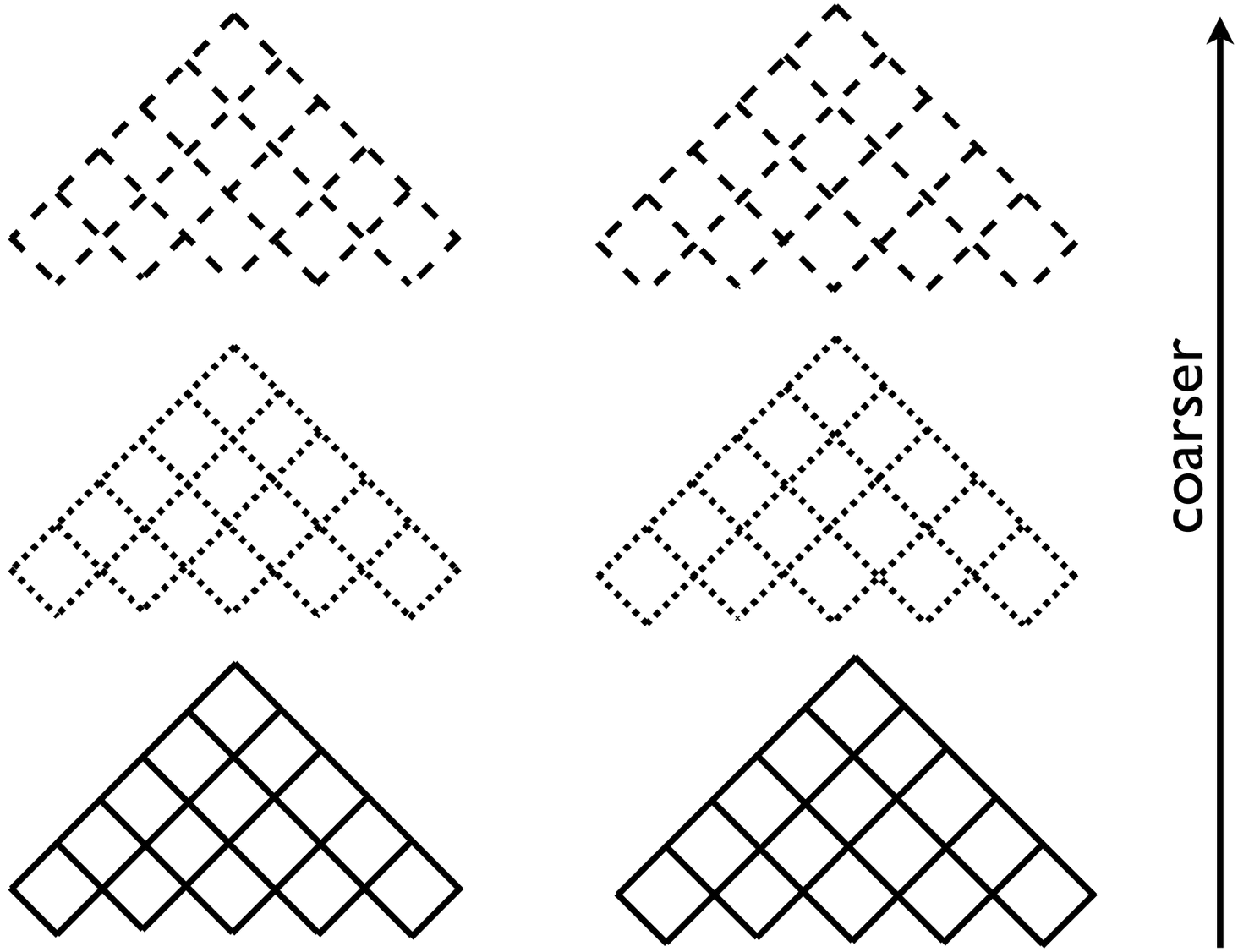
HA*



Agenda

inside

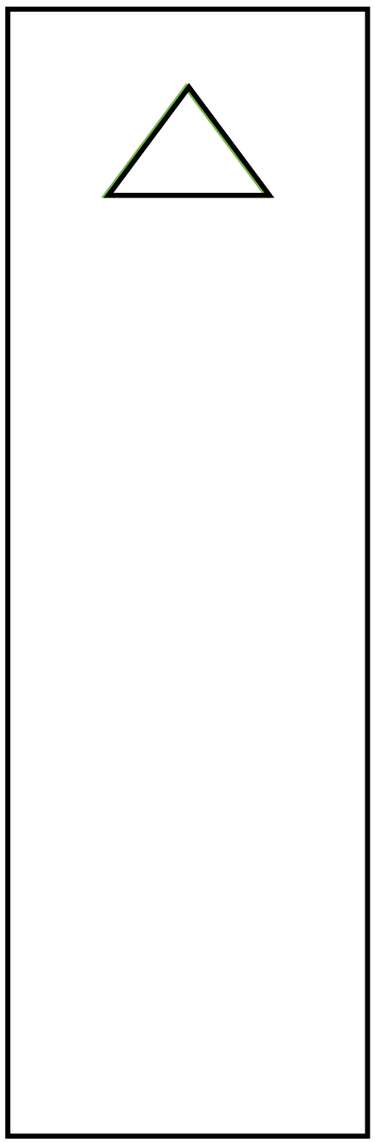
outside



coarser

Charts

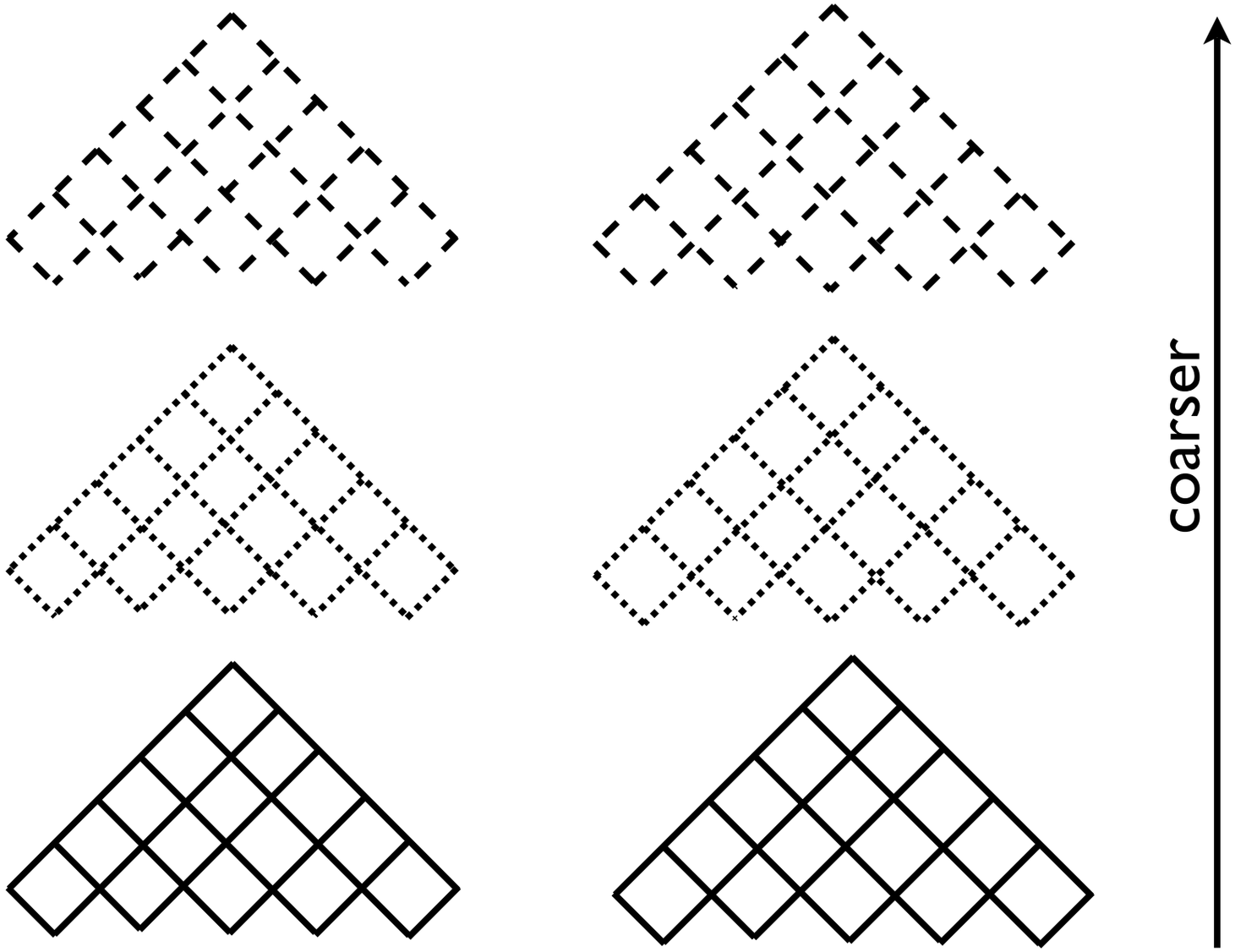
HA*



Agenda

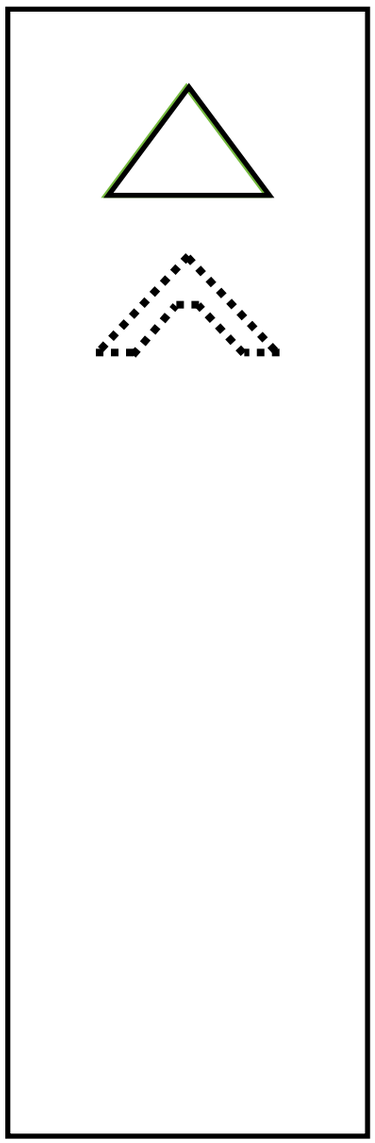
inside

outside



Charts

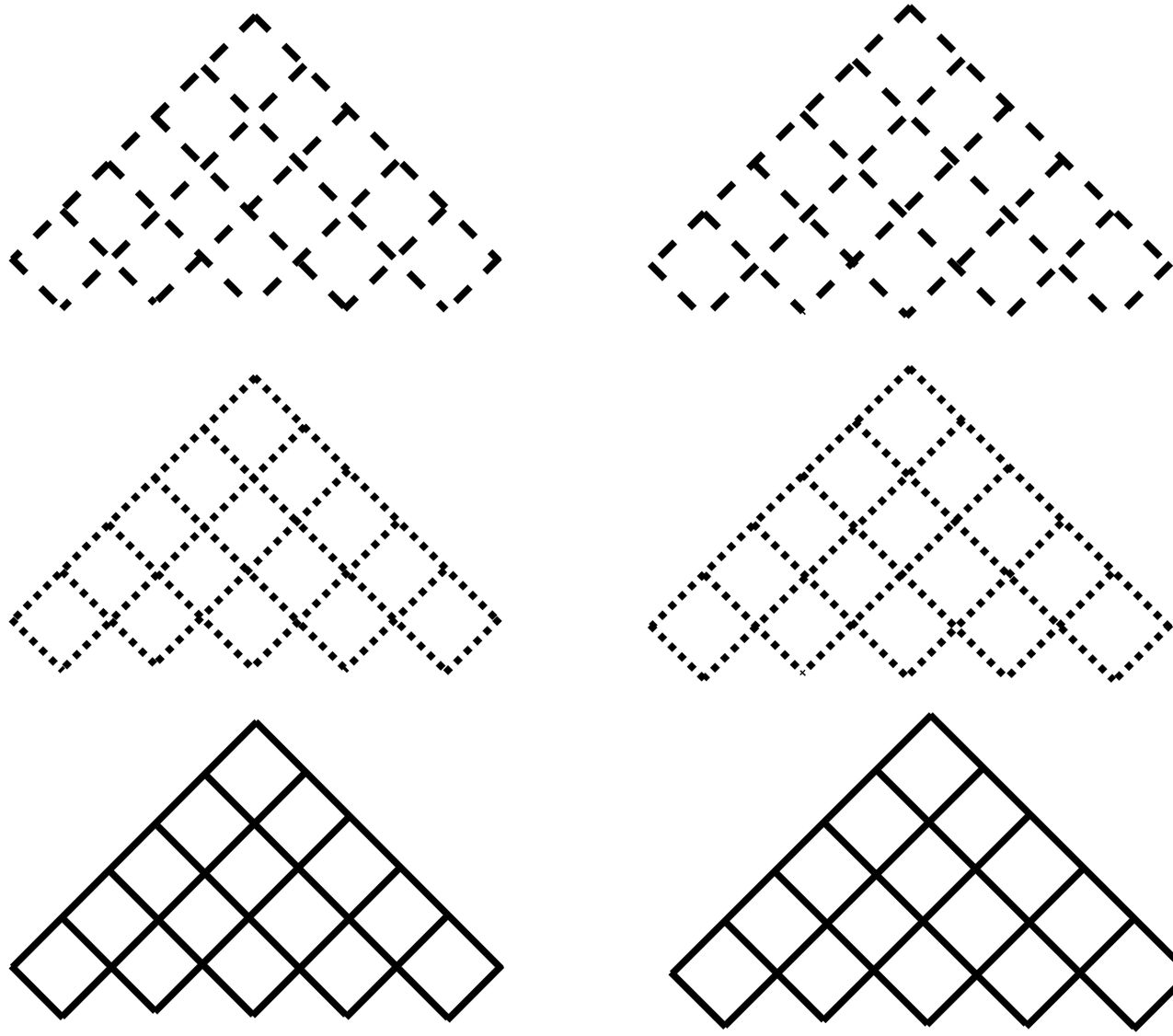
HA*



Agenda

inside

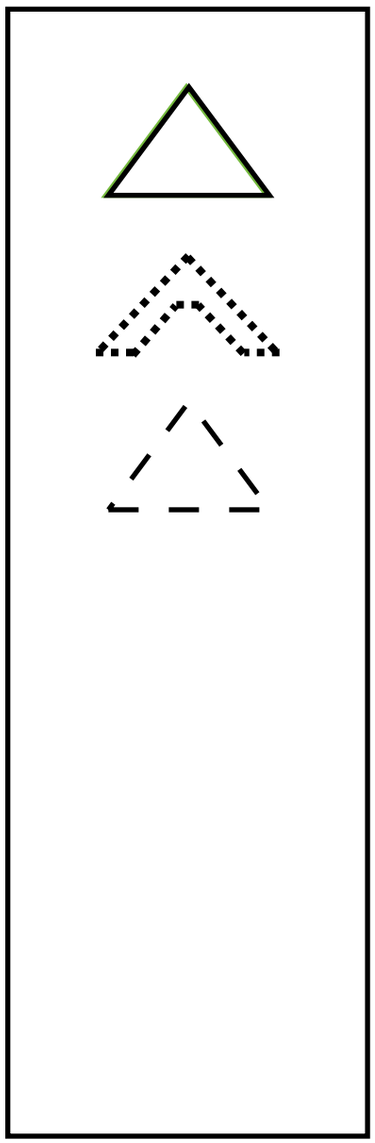
outside



coarser

Charts

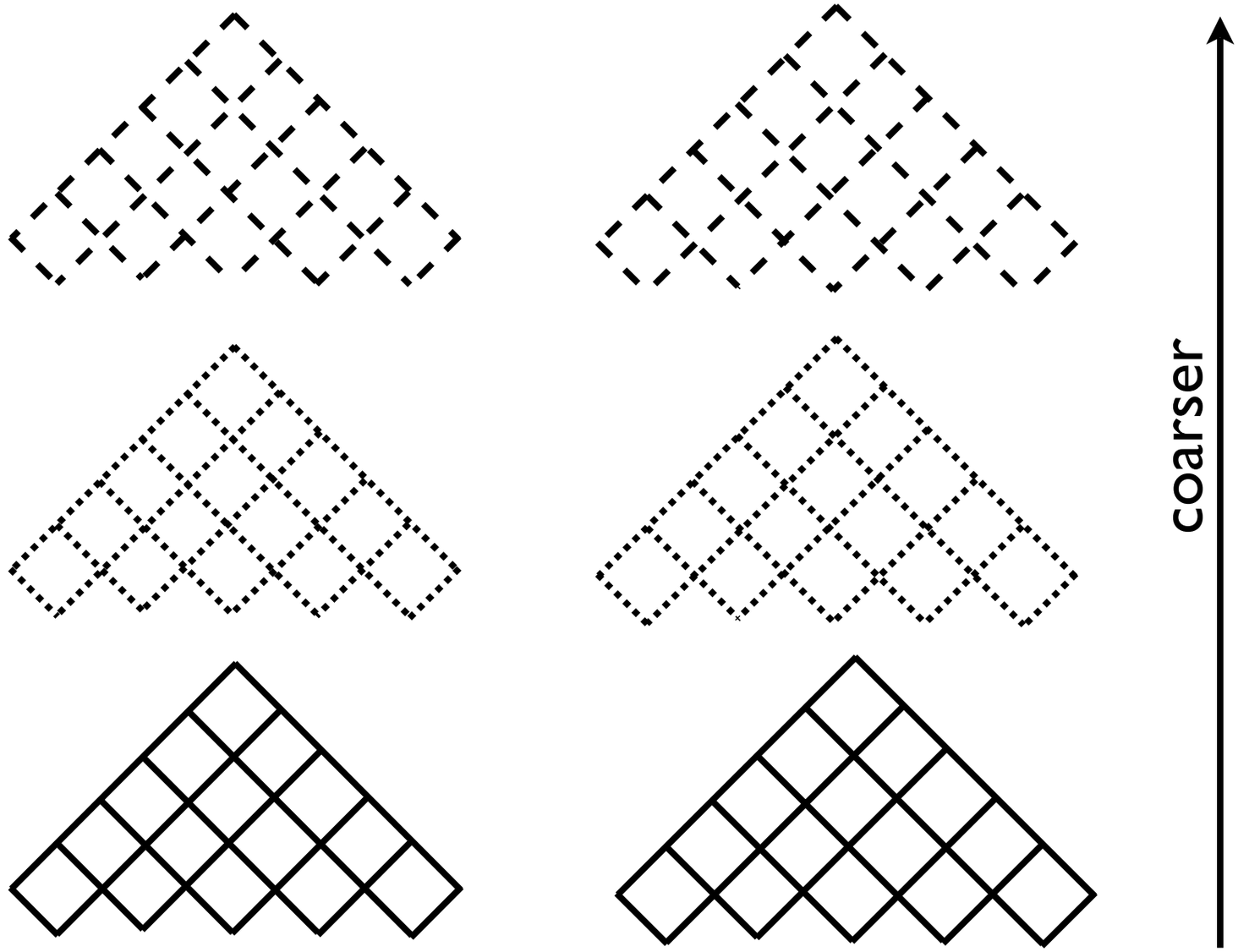
HA*



Agenda

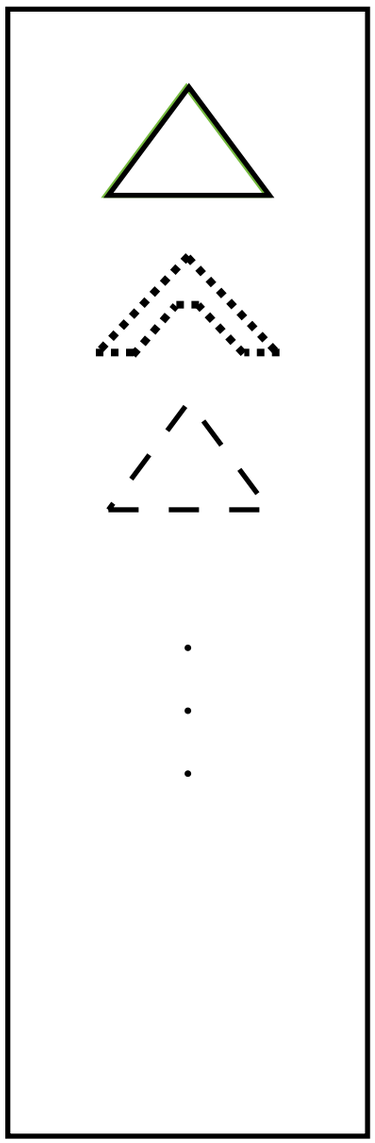
inside

outside



Charts

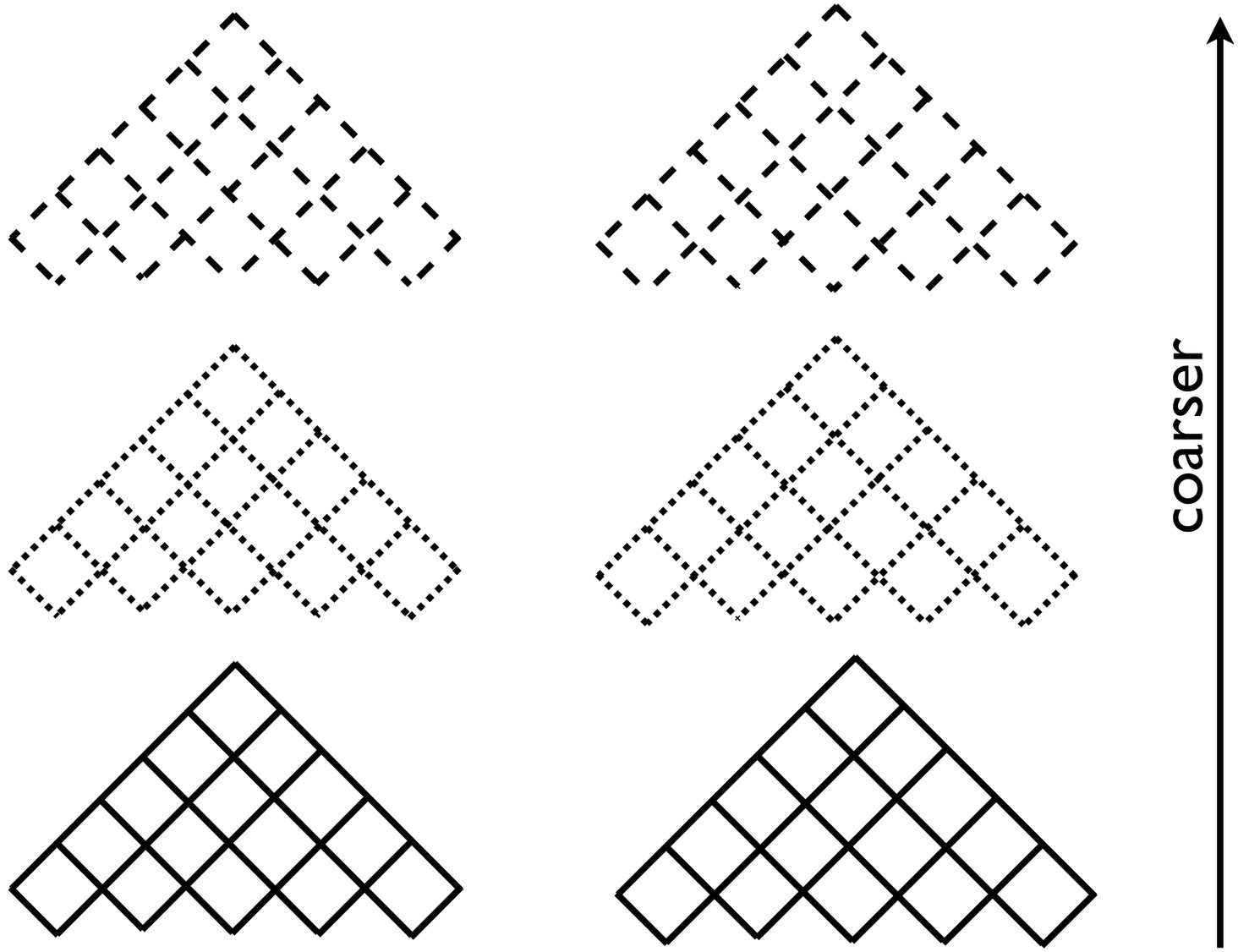
HA*



Agenda

inside

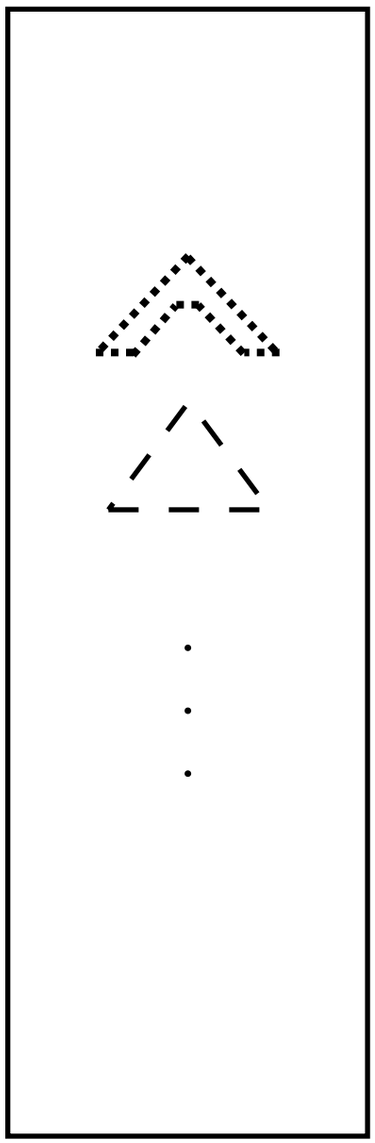
outside



Charts

University of California
Berkeley

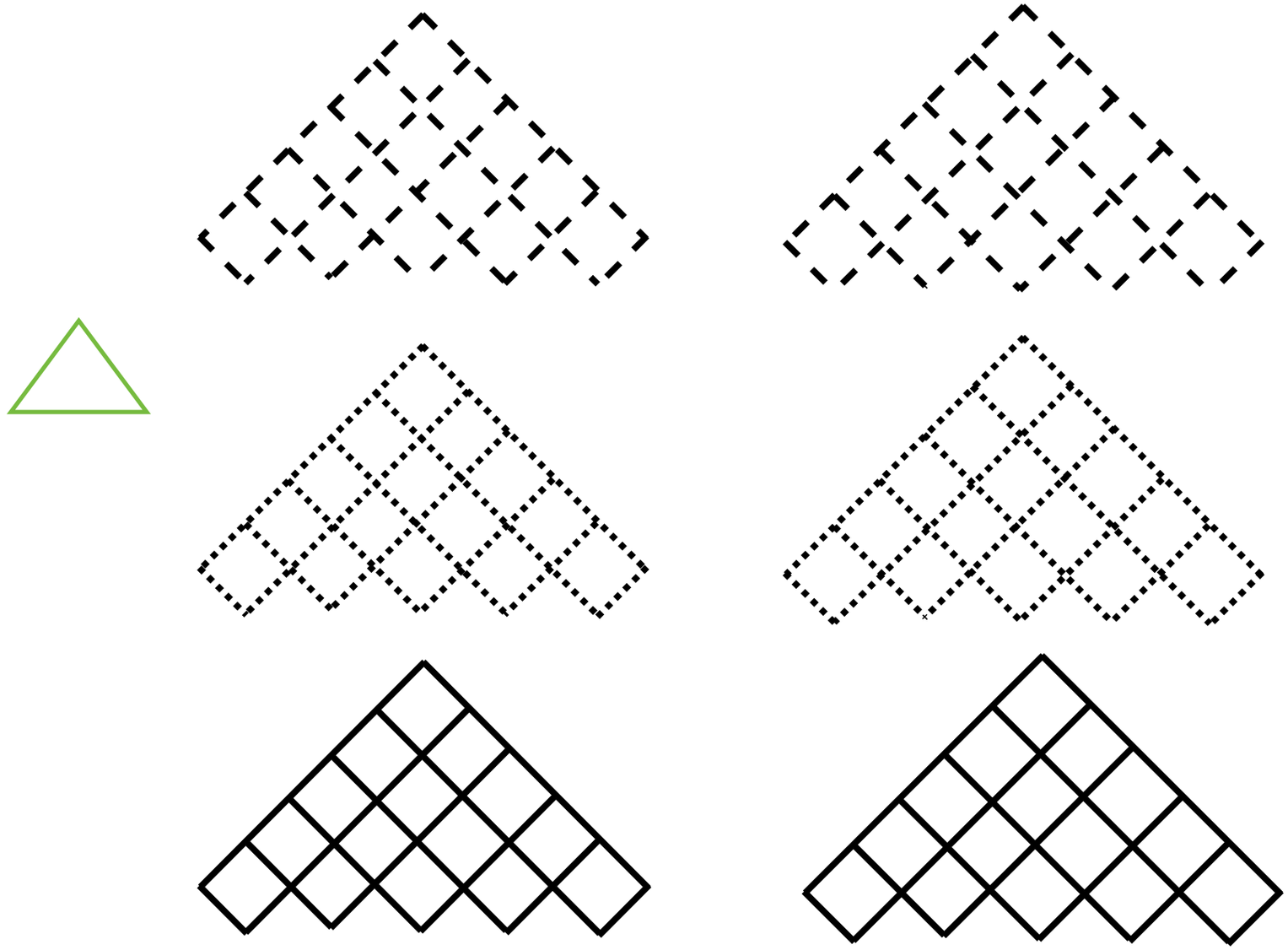
HA*



Agenda

inside

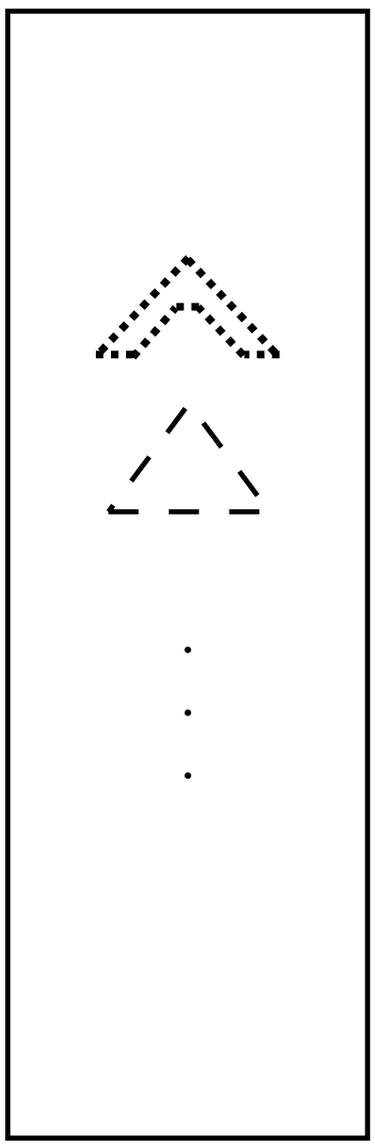
outside

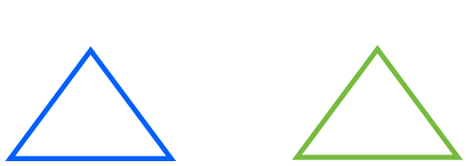


coarser

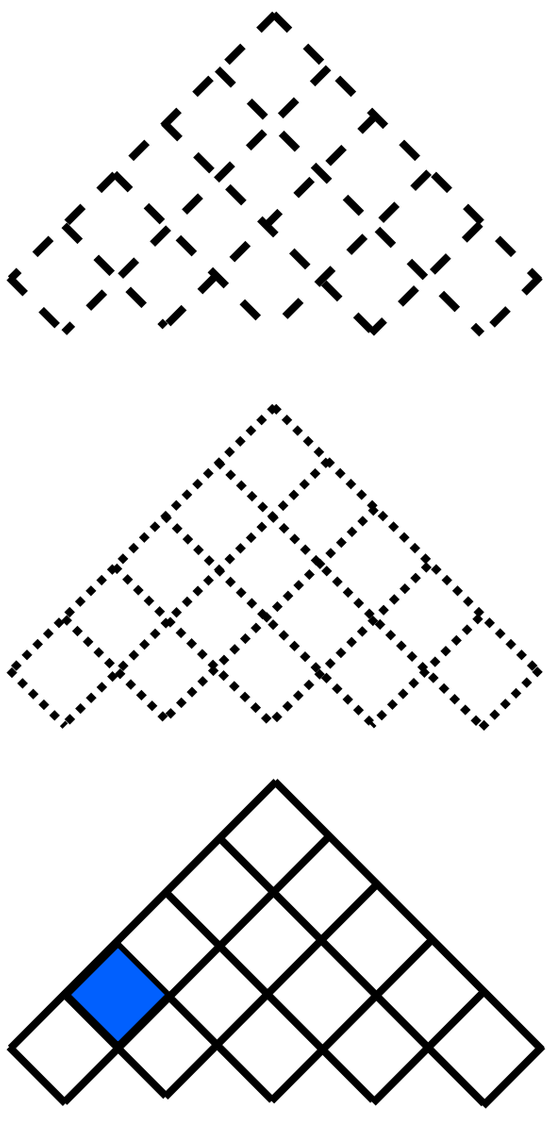
Charts

HA*

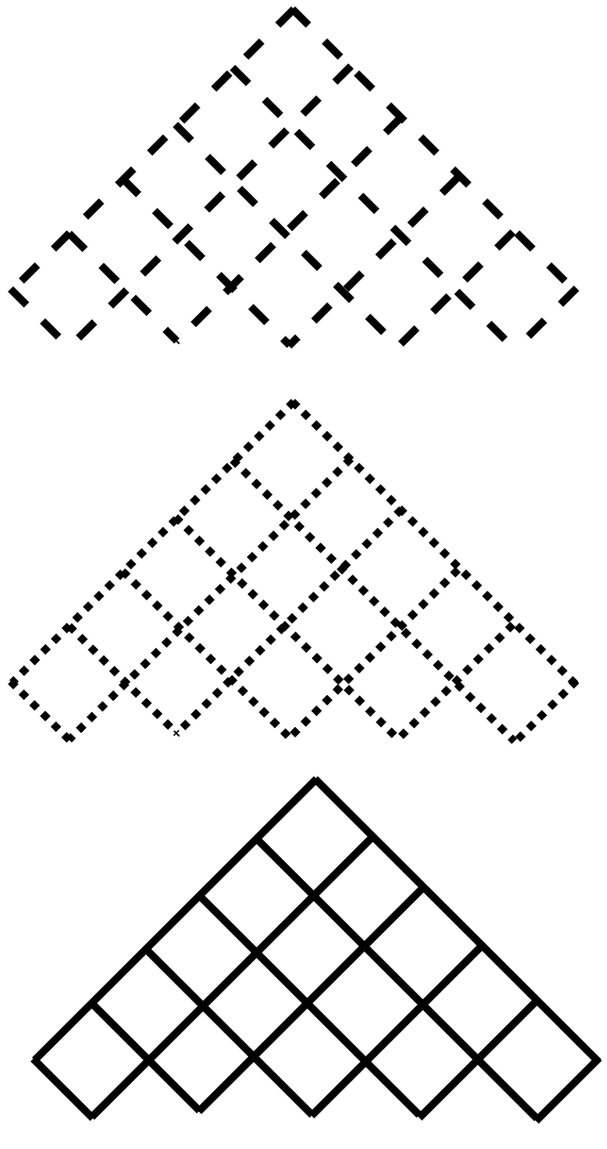




inside



outside

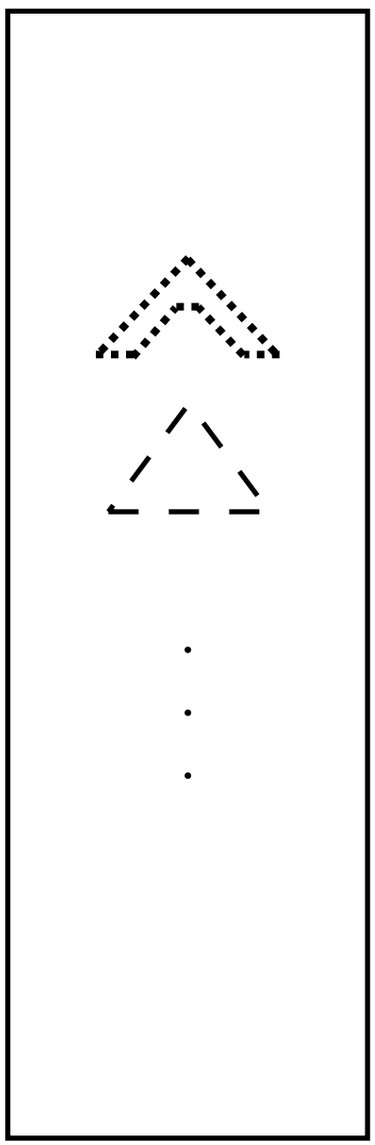


coarser

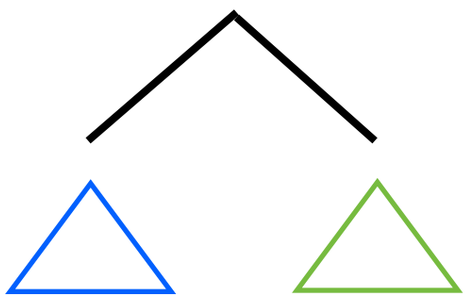
Agenda

Charts

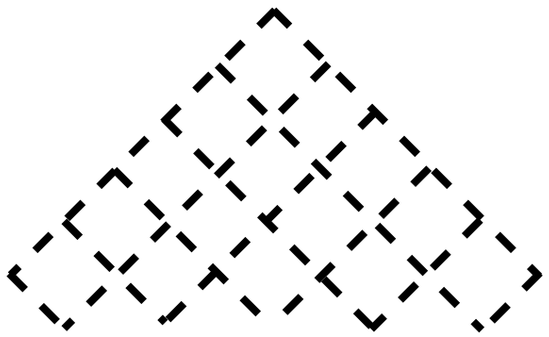
HA*

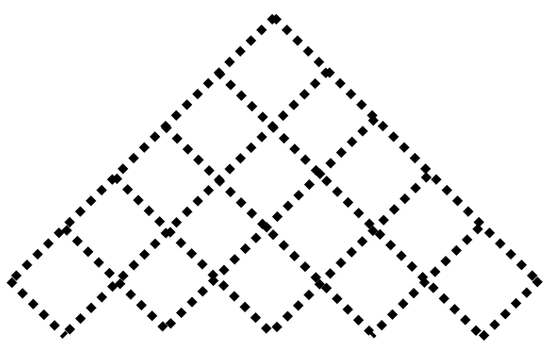


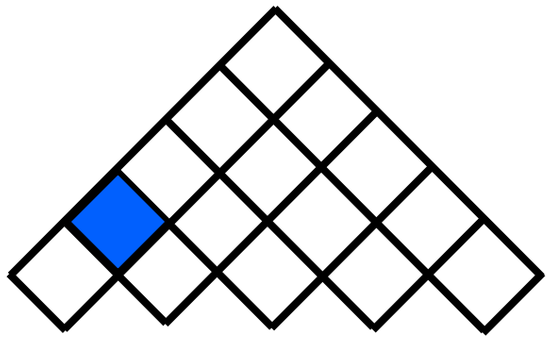
Agenda



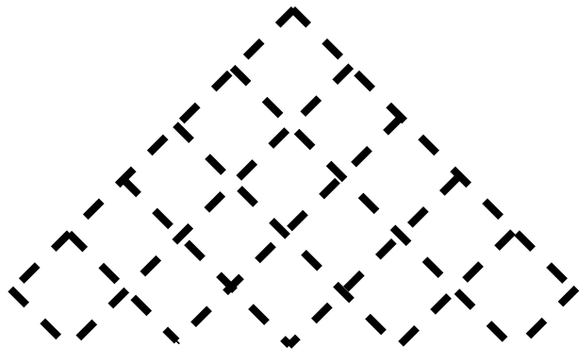
inside

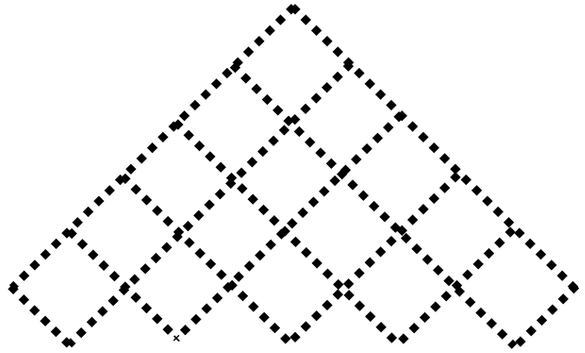


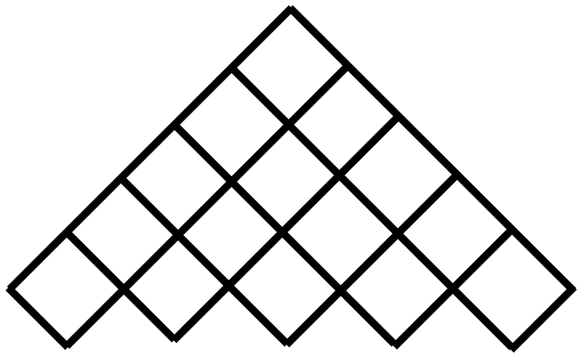




outside



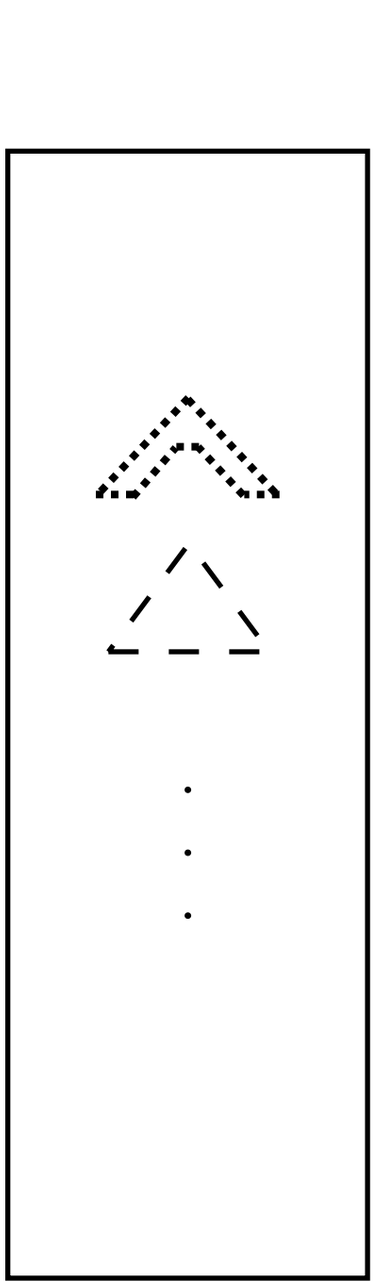




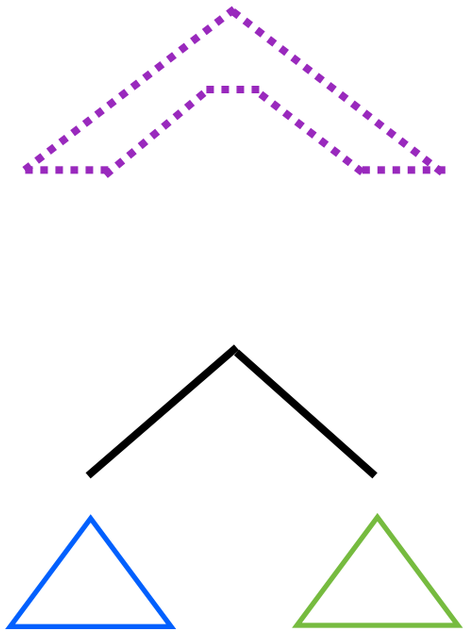
coarser

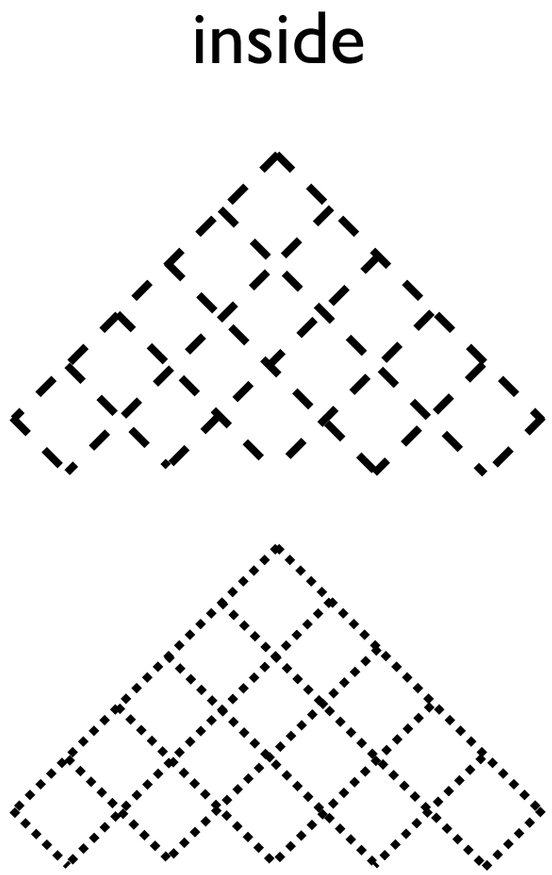
Charts

HA*

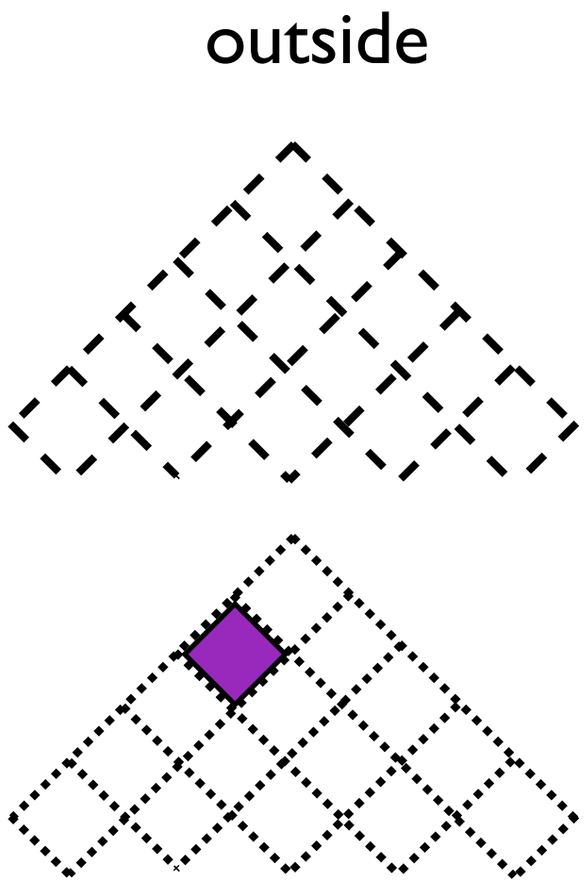


Agenda

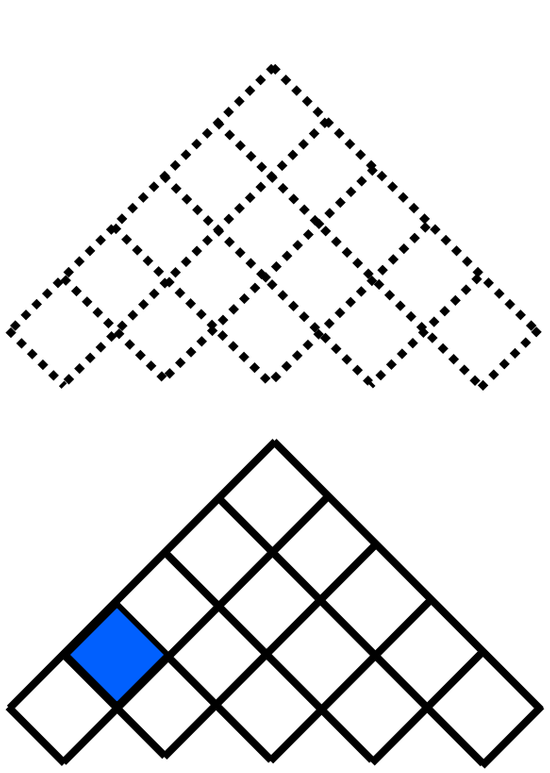




inside



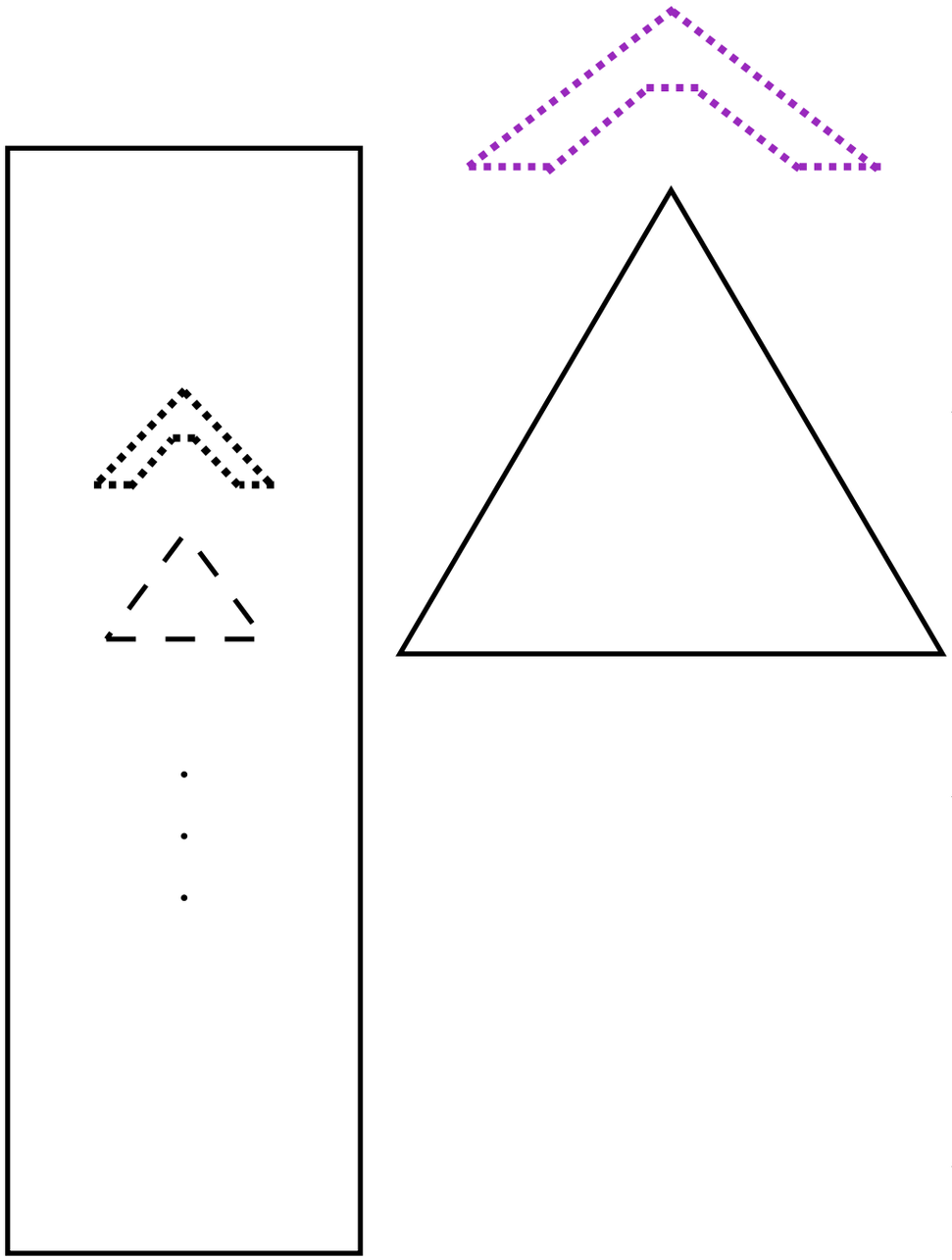
outside



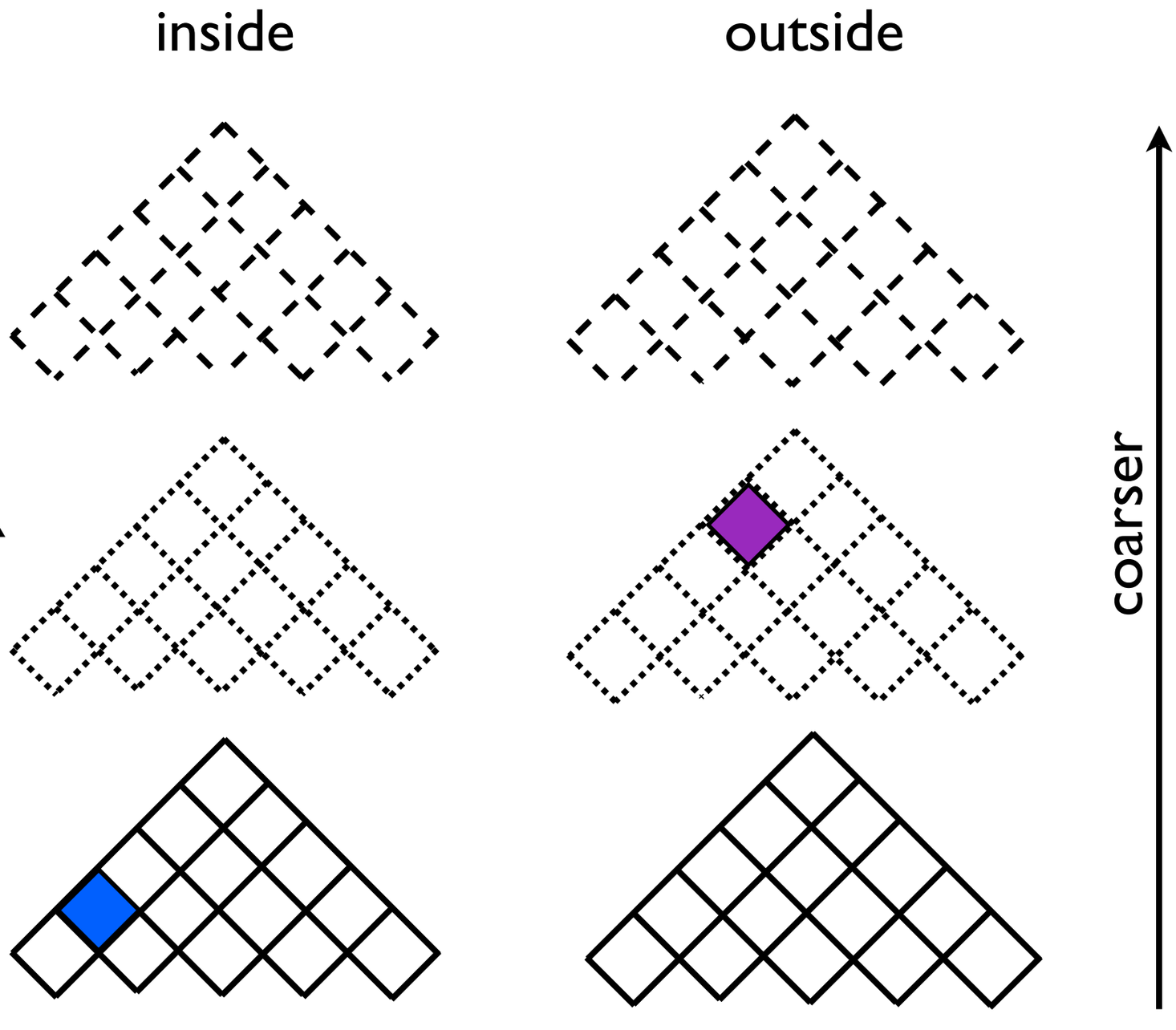
Charts

coarser

HA*

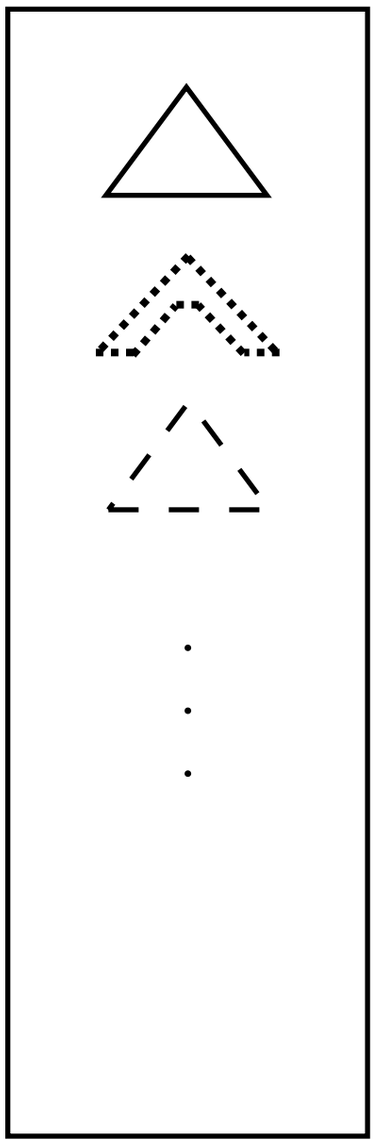


Agenda

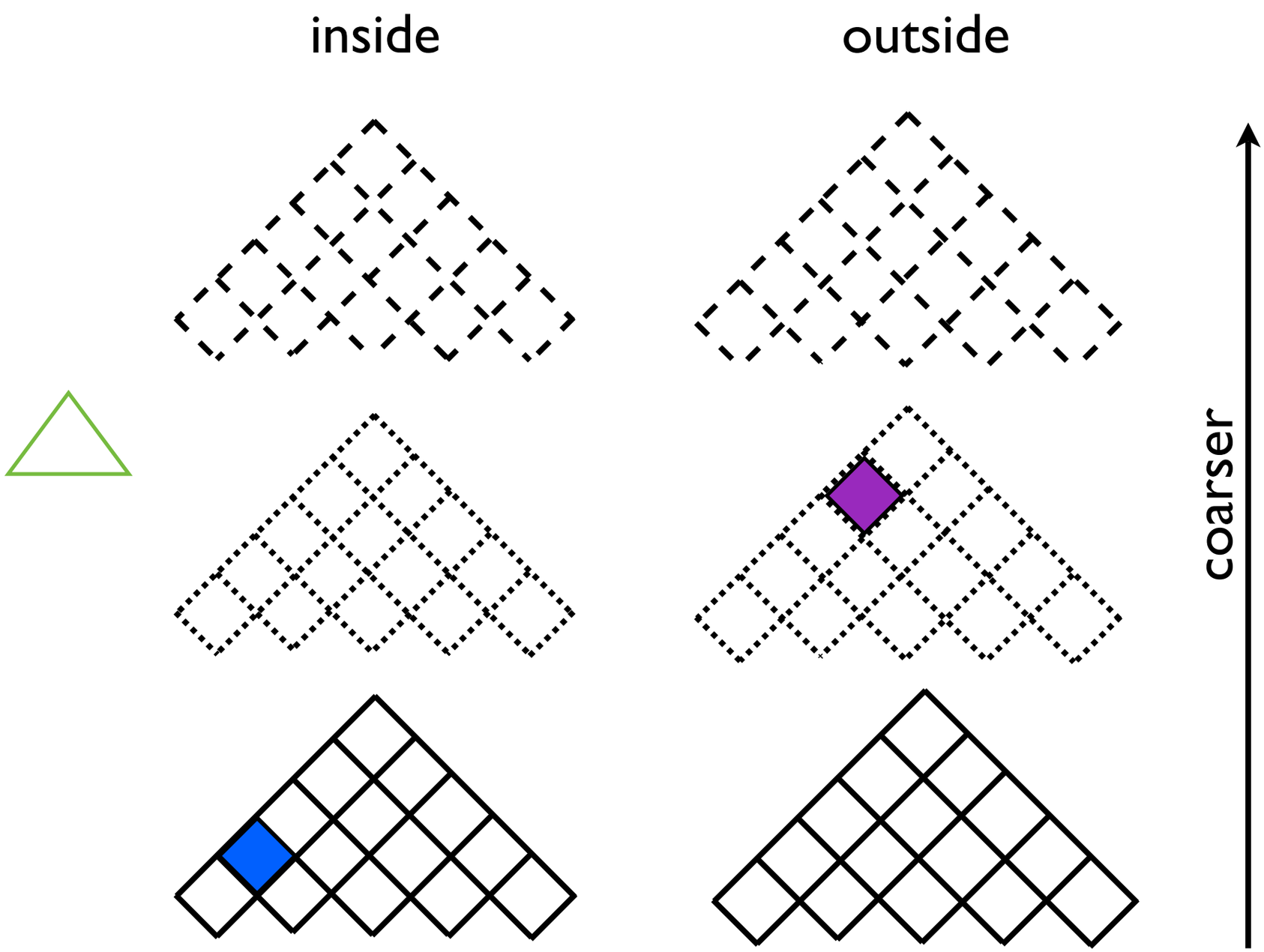


Charts

HA*

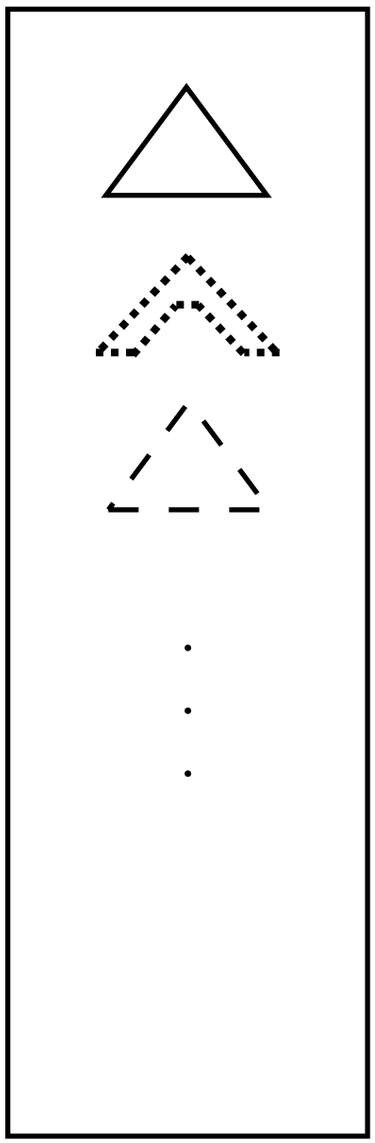


Agenda



Charts

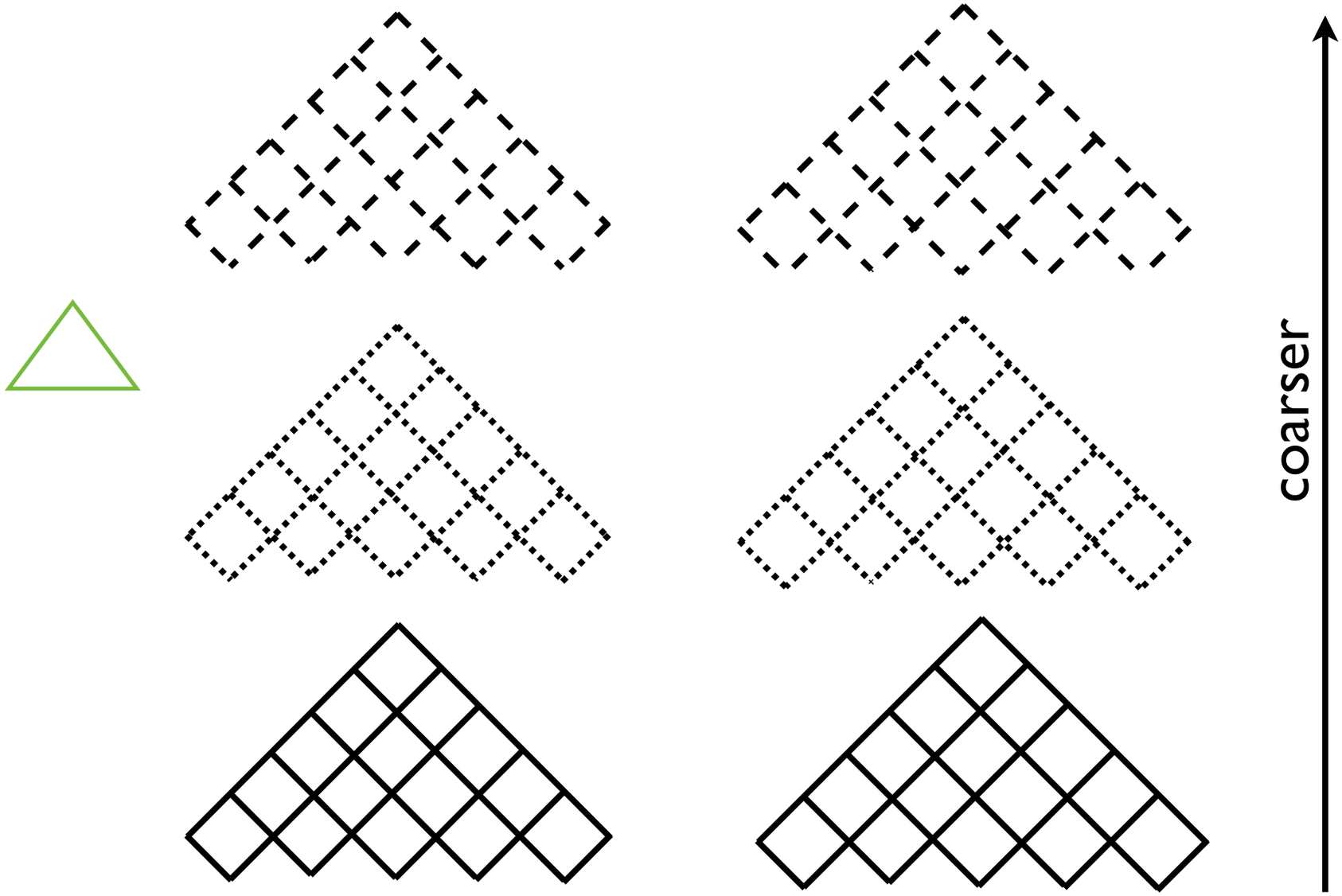
HA*



Agenda

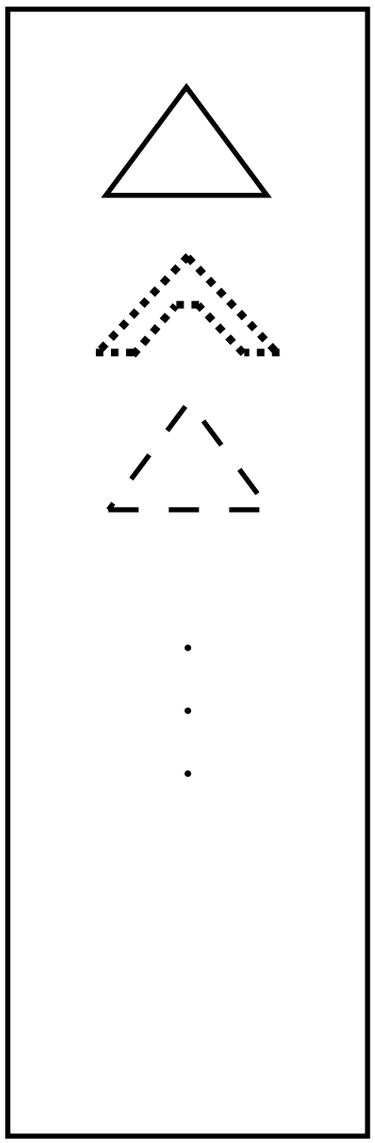
inside

outside



Charts

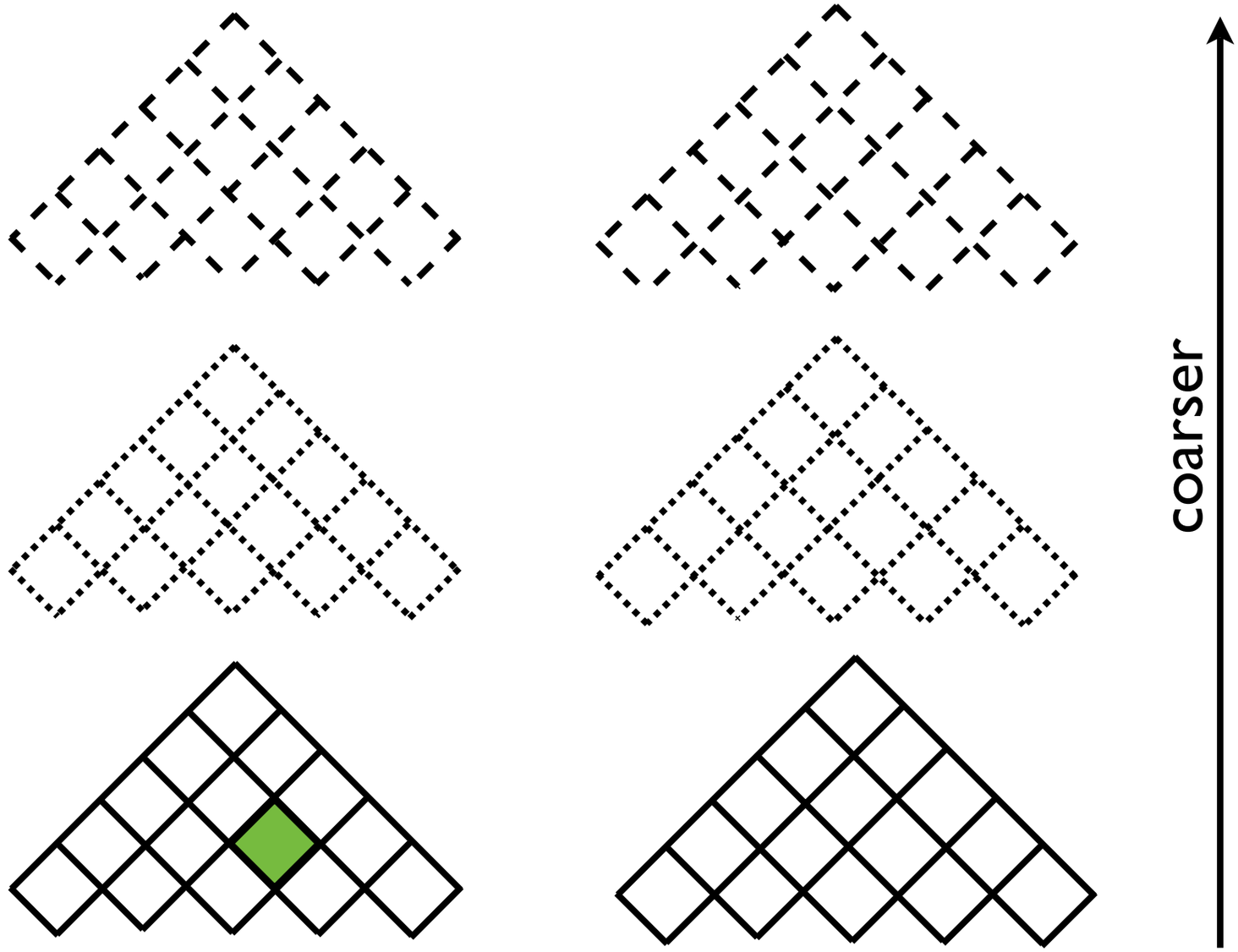
HA*



Agenda

inside

outside



Charts

Coarse-to-Fine

- Prune edges in fine grammar based on posteriors from coarse grammar

Coarse-to-Fine

- Prune edges in fine grammar based on posteriors from coarse grammar

- We use Viterbi posteriors for pruning (Petrov and Klein 2007)

Coarse-to-Fine

- Prune edges in fine grammar based on posteriors from coarse grammar

- We use Viterbi posteriors for pruning (Petrov and Klein 2007)

$$\beta'(e) + \alpha'(e) \leq \text{threshold}$$

Agenda-Based CTF

- (Hierarchical) CTF can also be thought of as an instance of agenda-based parsing with

$$priority(e) = \begin{cases} \beta(e) & \beta'(e) + \alpha'(e) \leq \text{threshold} \\ \infty & \text{otherwise} \end{cases}$$

Agenda-Based CTF

- (Hierarchical) CTF can also be thought of as an instance of agenda-based parsing with

$$priority(e) = \begin{cases} \beta(e) & \beta'(e) + \alpha'(e) \leq \text{threshold} \\ \infty & \text{otherwise} \end{cases}$$

- This reformulation makes architectures directly comparable

HA* vs. HCTF Qualitatively

HA*	HCTF

University of California
C A L
N L P
Berkeley

HA* vs. HCTF Qualitatively

HA*

- optimal

HCTF

- makes search errors

HA* vs. HCTF Qualitatively

HA*

- optimal
- uses coarse grammars to prioritize search

HCTF

- makes search errors
- uses coarse grammars to prune search

HA* vs. HCTF Qualitatively

HA*

- optimal
- uses coarse grammars to prioritize search
- speed determined by tightness of heuristic

HCTF

- makes search errors
- uses coarse grammars to prune search
- speed determined by threshold

HA* vs. HCTF Qualitatively

HA*

- optimal
- uses coarse grammars to prioritize search
- speed determined by tightness of heuristic
- min over rules

HCTF

- makes search errors
- uses coarse grammars to prune search
- speed determined by threshold
- average over rules

University of California
CALNLP
Berkeley

Experimental Setup #1

- Use the state-split grammars of Petrov *et al.* 2006
- Train on WSJ Sections 2-21, and use 6 split iterations, which creates 7 grammars

Experimental Setup #1

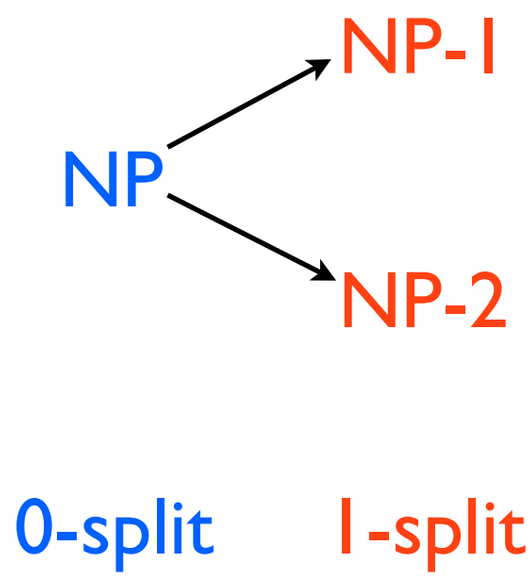
- Use the state-split grammars of Petrov *et al.* 2006
- Train on WSJ Sections 2-21, and use 6 split iterations, which creates 7 grammars

NP

0-split

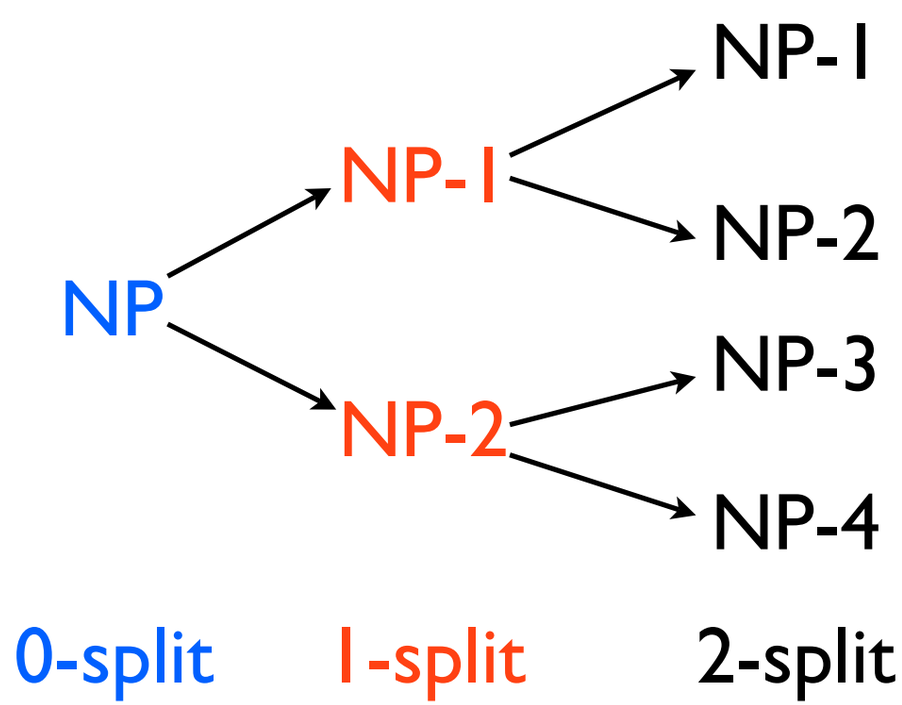
Experimental Setup #1

- Use the state-split grammars of Petrov *et al.* 2006
- Train on WSJ Sections 2-21, and use 6 split iterations, which creates 7 grammars



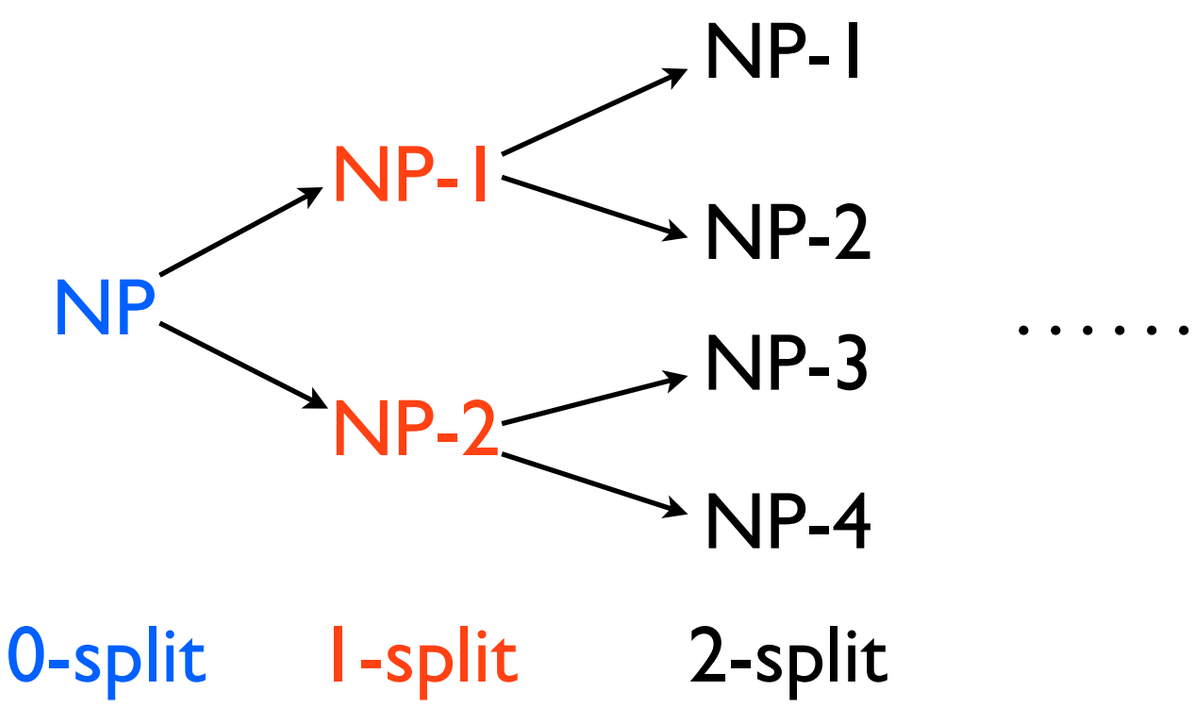
Experimental Setup #1

- Use the state-split grammars of Petrov *et al.* 2006
- Train on WSJ Sections 2-21, and use 6 split iterations, which creates 7 grammars

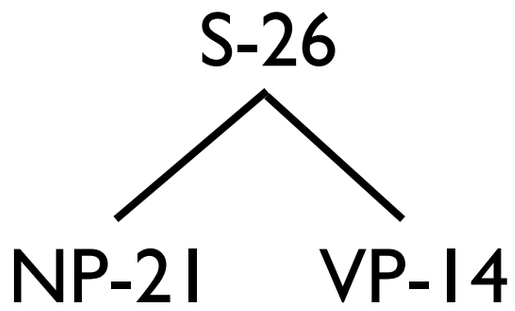


Experimental Setup #1

- Use the state-split grammars of Petrov *et al.* 2006
- Train on WSJ Sections 2-21, and use 6 split iterations, which creates 7 grammars

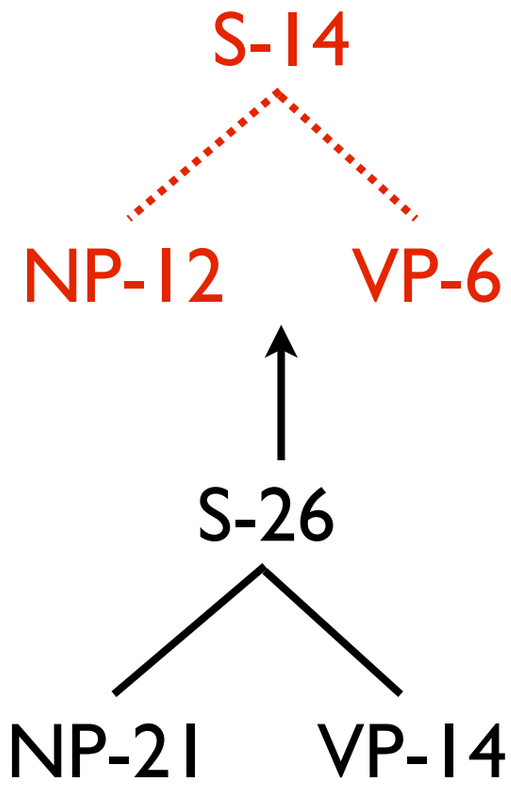


State-Split Projections



6-split

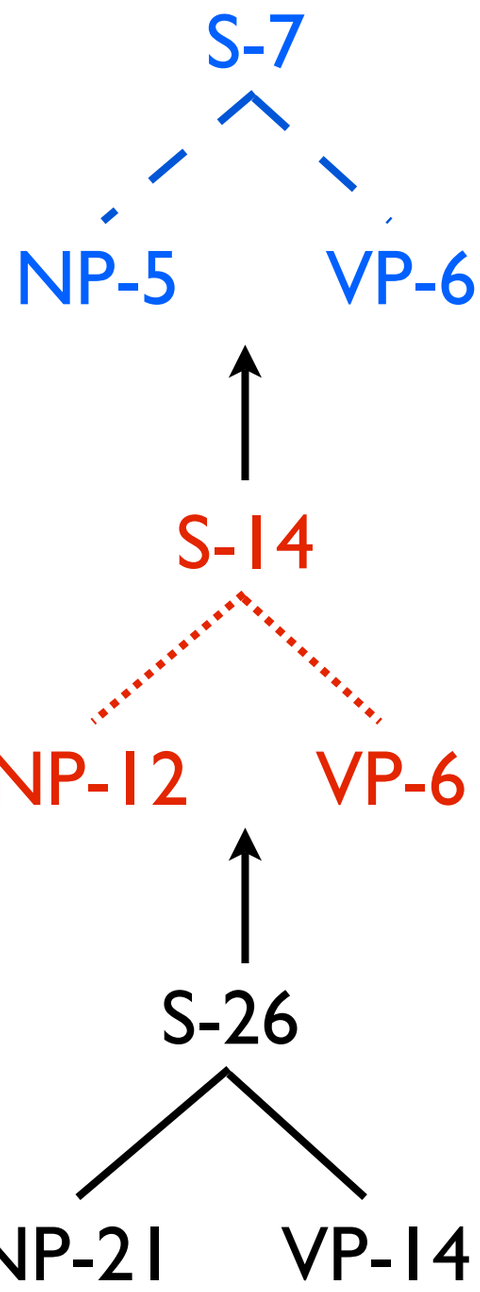
State-Split Projections



5-split

6-split

State-Split Projections

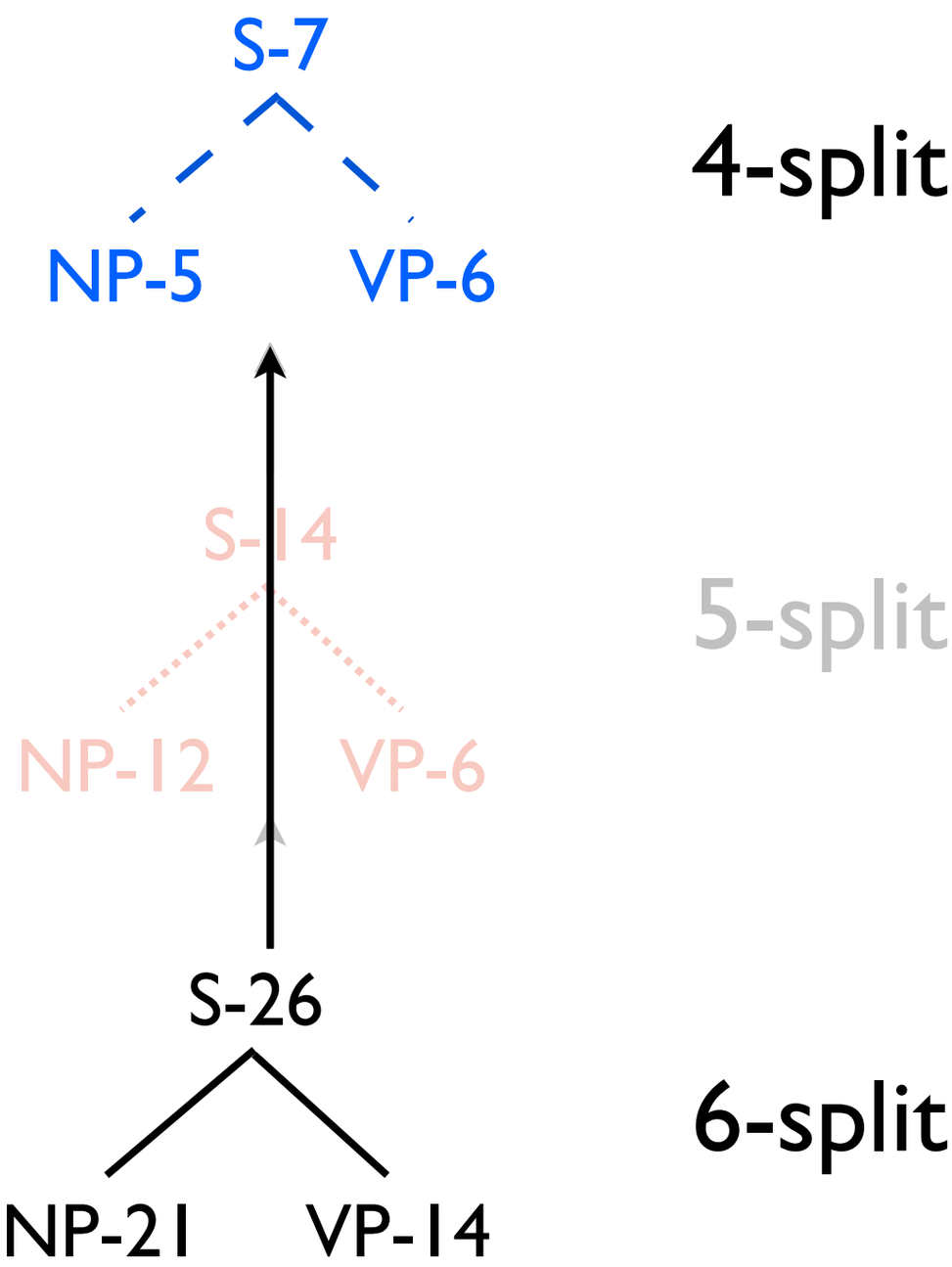


4-split

5-split

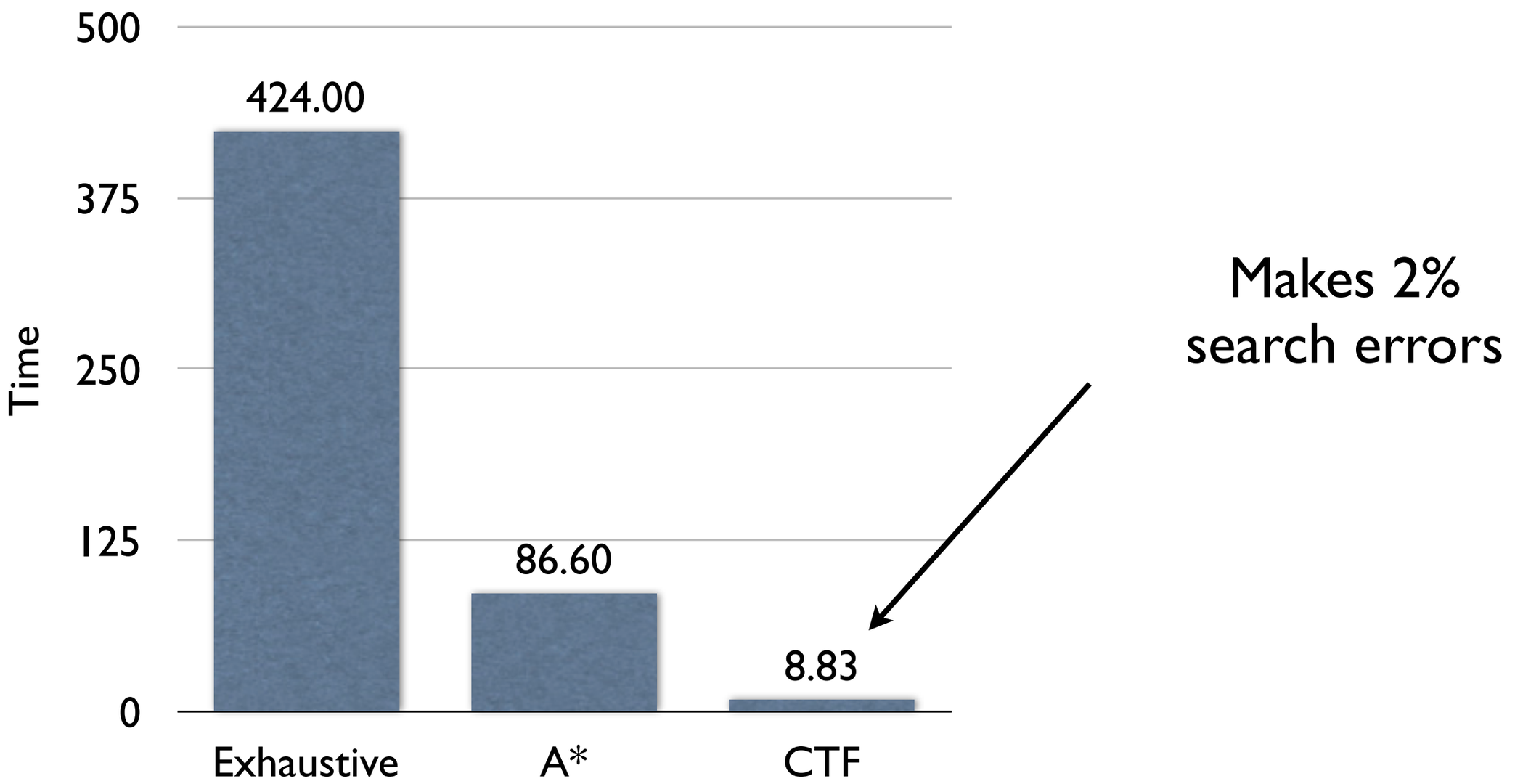
6-split

State-Split Projections



One-Level CTF vs. A*

- Only one coarse grammar (the 3-split)
- CTF is faster than A*, but makes search errors



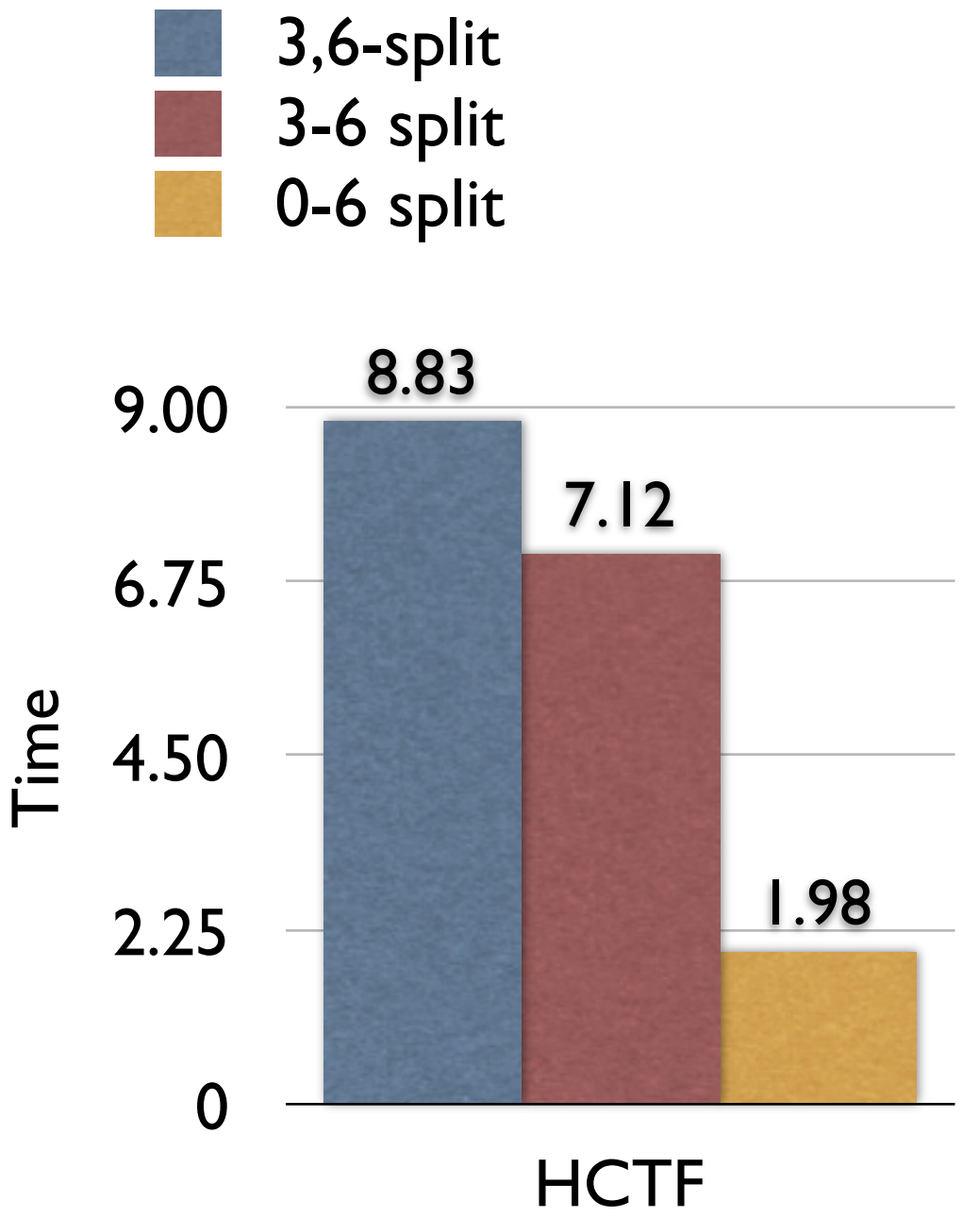
University of California
C A L
N L P
Berkeley

Hierarchies

- How do HCTF and HA* scale with size of hierarchy?

Hierarchies

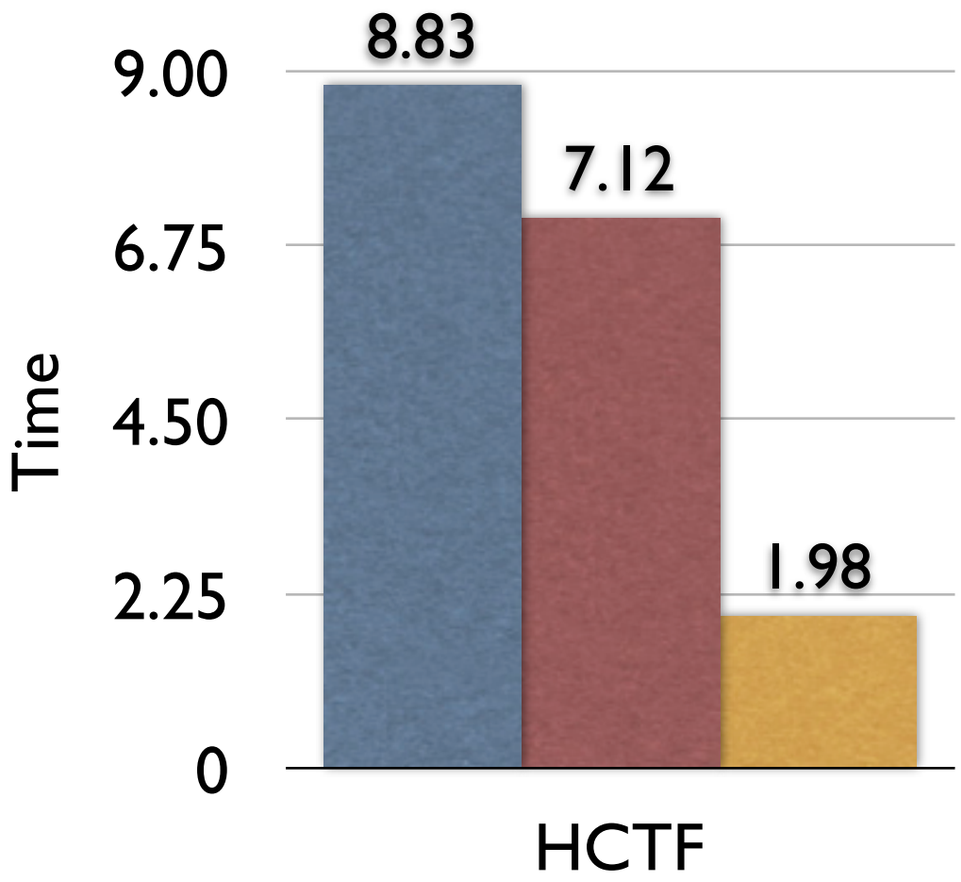
- How do HCTF and HA* scale with size of hierarchy?

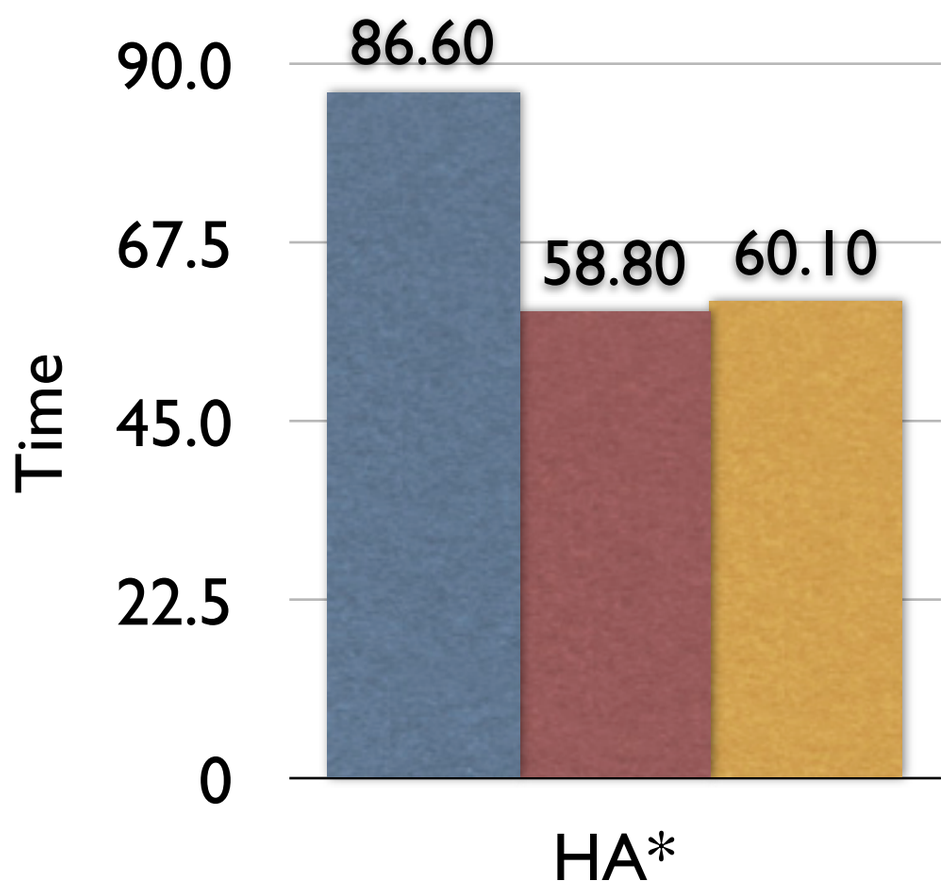


Hierarchies

- How do HCTF and HA* scale with size of hierarchy?

3,6-split
3-6 split
0-6 split





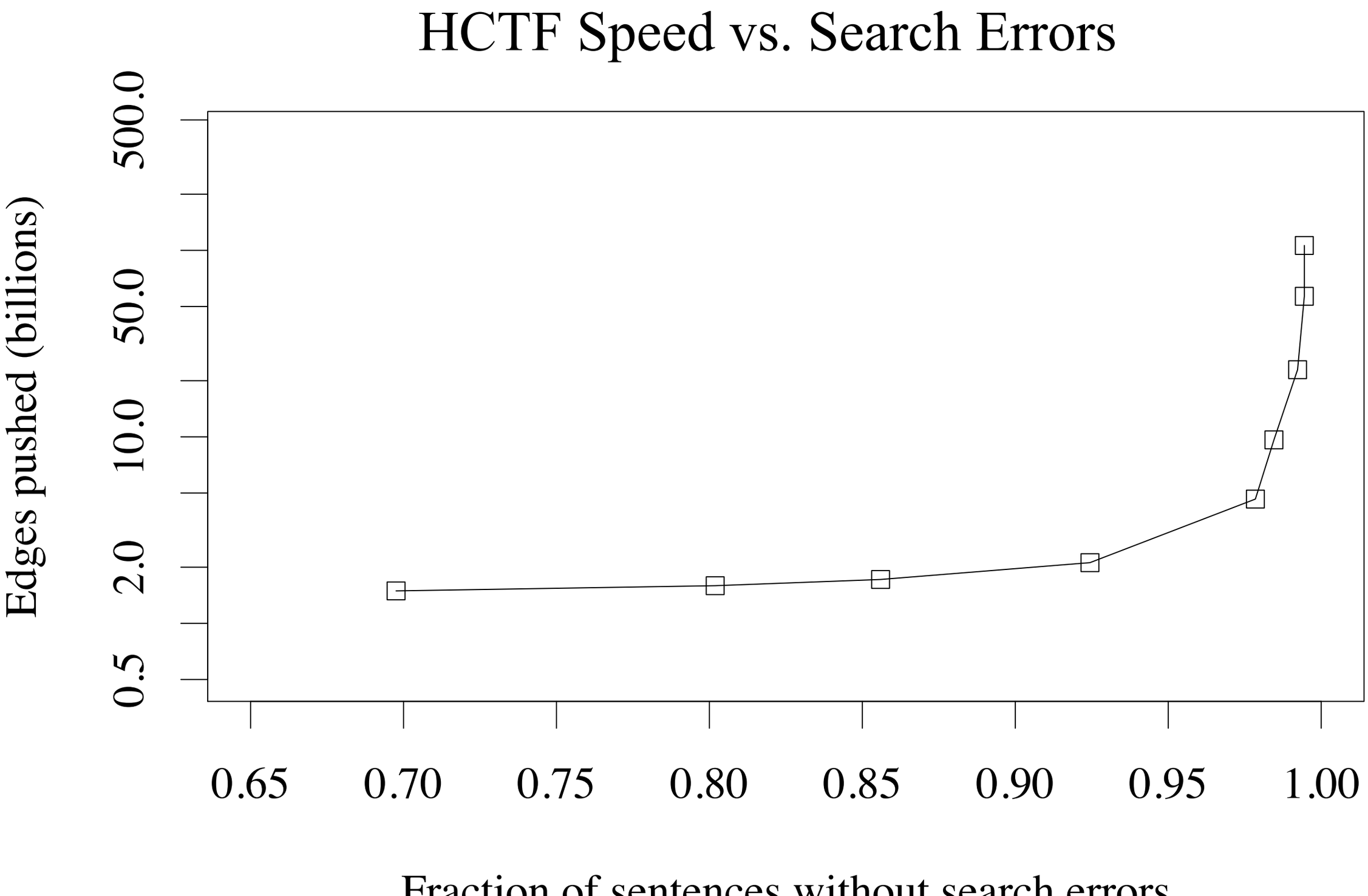
University of California
Berkeley

Why A*?

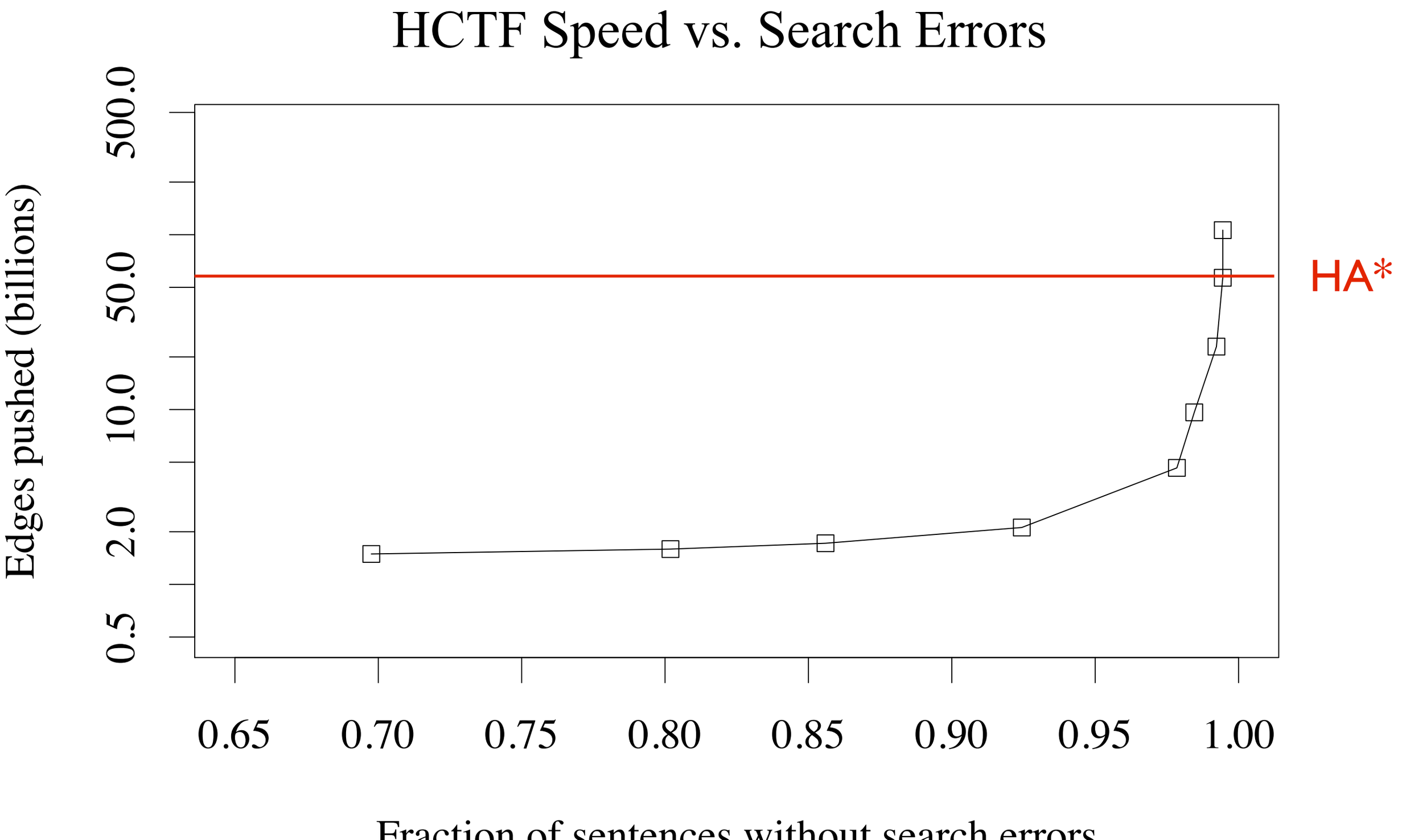
- CTF is faster, and extends to hierarchies nicely, so why A*?

1. If you really don't want to make search errors

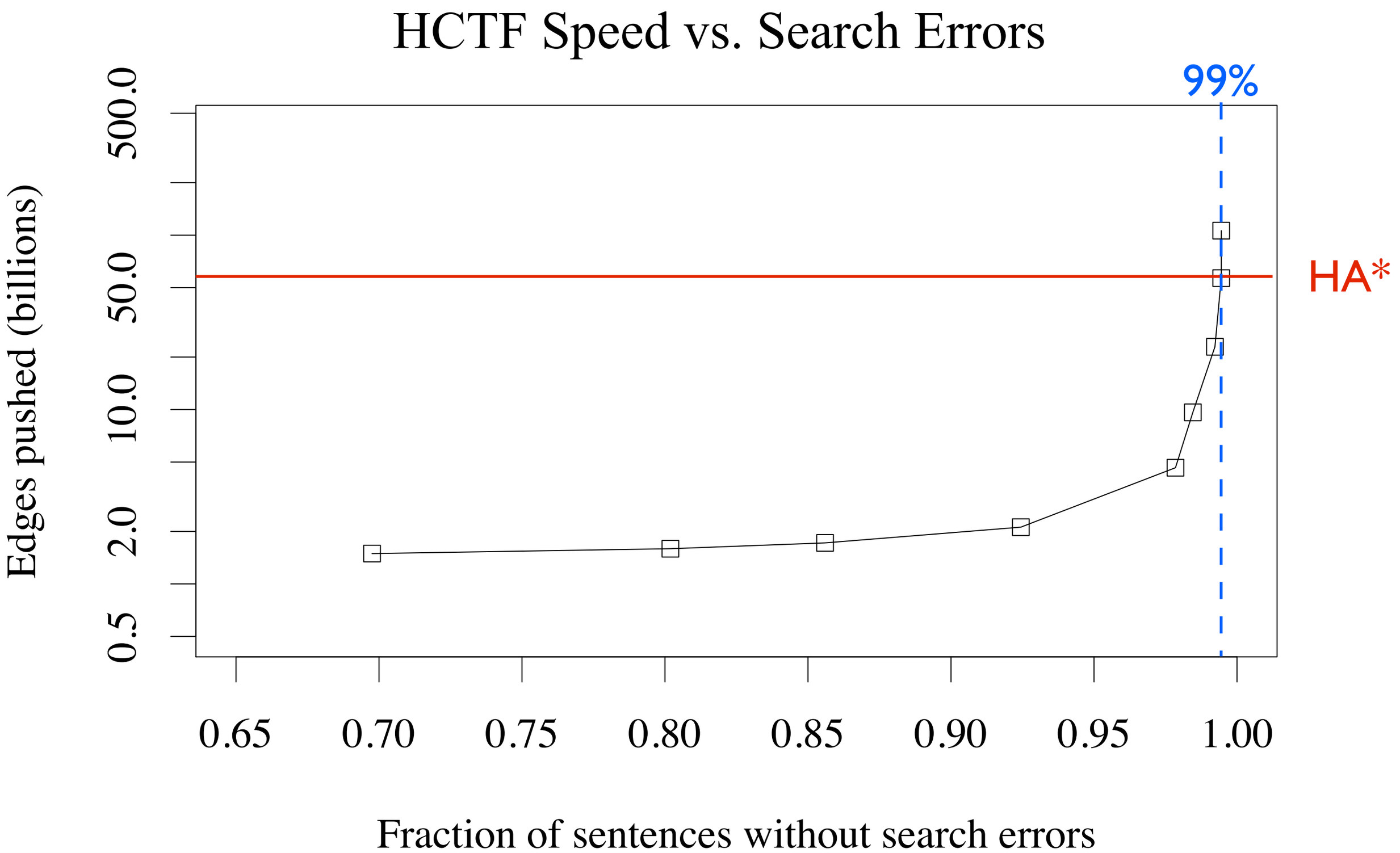
Cost of Optimality: State-Split Grammars



Cost of Optimality: State-Split Grammars



Cost of Optimality: State-Split Grammars



Why A*?

- CTF is faster, and extends to hierarchies nicely, so why A*?

1. If you really don't want to make search errors

2. For some problems, we can find very efficient, tight heuristics

- In this case, A* is very fast

Experimental Setup #2

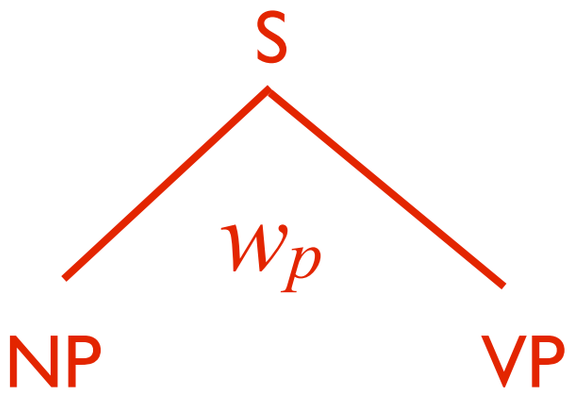
- We use the factored lexicalized grammar of Klein and Manning (2003)

- They construct a lexicalized grammar as the cross-product of a dependency grammar and PCFG

Experimental Setup #2

- We use the factored lexicalized grammar of Klein and Manning (2003)

- They construct a lexicalized grammar as the cross-product of a dependency grammar and PCFG

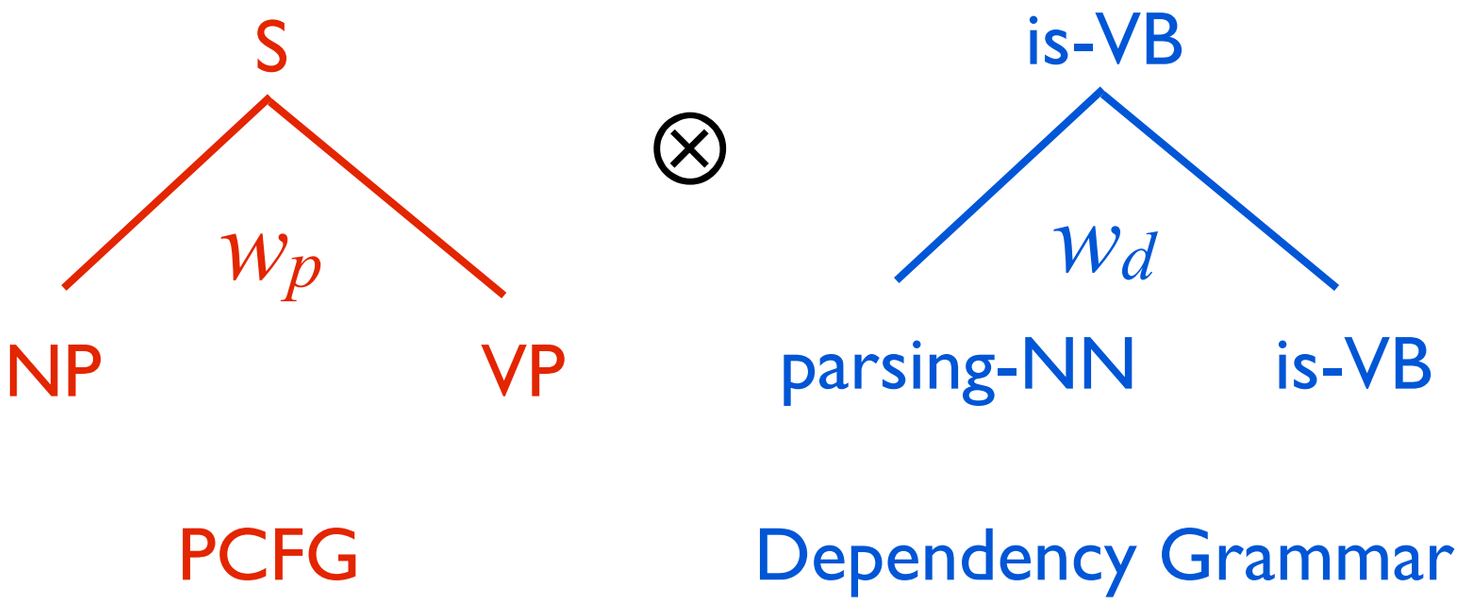


PCFG

Experimental Setup #2

- We use the factored lexicalized grammar of Klein and Manning (2003)

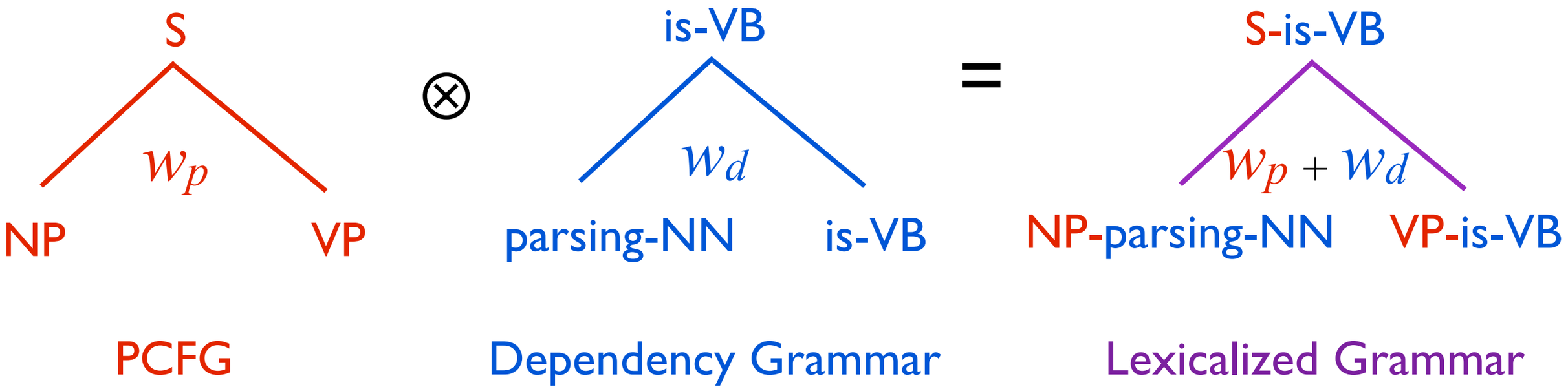
- They construct a lexicalized grammar as the cross-product of a dependency grammar and PCFG



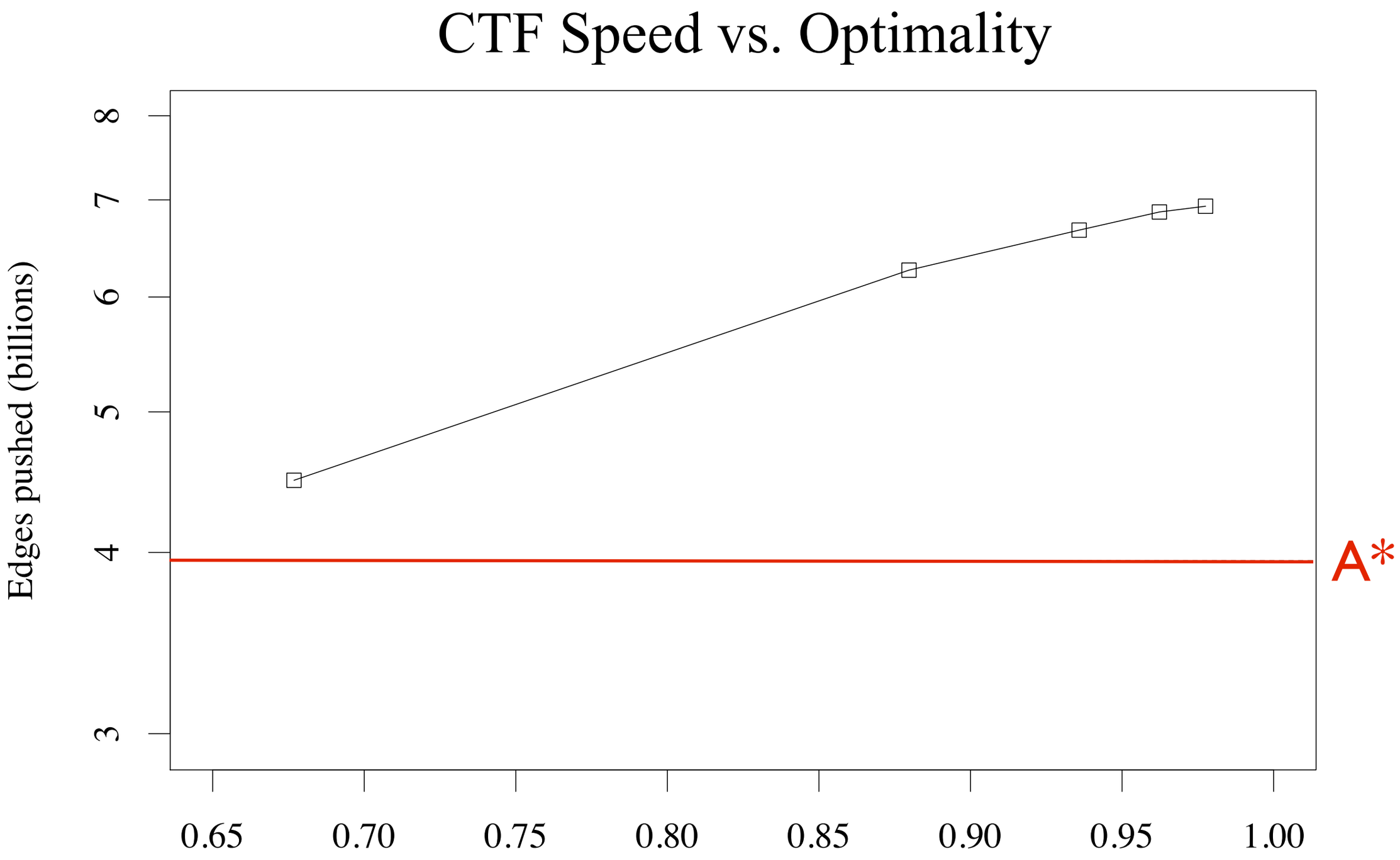
Experimental Setup #2

- We use the factored lexicalized grammar of Klein and Manning (2003)

- They construct a lexicalized grammar as the cross-product of a dependency grammar and PCFG



Cost of Optimality: Lexicalized Grammar



Conclusions

- Coarse-to-Fine is much faster for reasonable number of search errors
- Hierarchical Coarse-to-Fine effectively exploits multilevel hierarchies, Hierarchical A* does not
- Hierarchical A* is the right choice if
  - optimality is desired
  - heuristics are very tight

Thank you