University of Oslo : Department of Informatics


# *INF4820: Algorithms for Artificial Intelligence and Natural Language Processing*

## Context-Free Grammars & Parsing

Stephan Oepen & Murhaf Fares

Language Technology Group (LTG)

November 3, 2016

# Overview


## Last Time

- Sequence Labeling
- Dynamic programming
- Viterbi algorithm
- Forward algorithm

# Overview

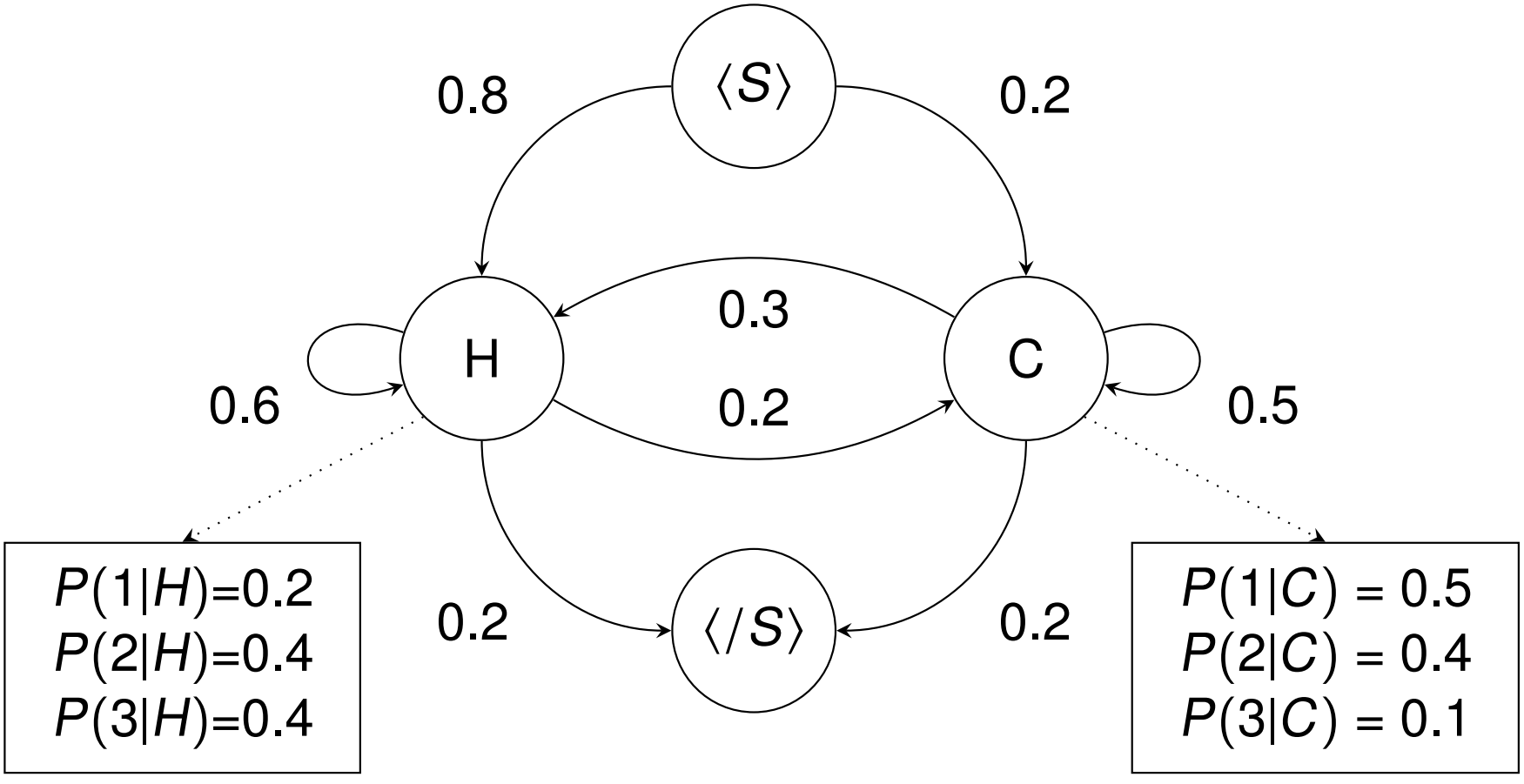

## Last Time

- Sequence Labeling
- Dynamic programming
- Viterbi algorithm
- Forward algorithm

## Today

- Grammatical structure
- Context-free grammar
- Treebanks
- Probabilistic CFGs

# Recall: Ice Cream and Global Warming



# Recall: Viterbi Algorithm


- To find the best state sequence, maximize:

  $P(s_1 \dots s_n | o_1 \dots o_n) = P(s_1|s_0)P(o_1|s_1)P(s_2|s_1)P(o_2|s_2)\dots$
- The value we cache at each step:

$$v_i(s) = \max_{k=1}^{L} [v_{i-1}(k) \cdot P(s|k) \cdot P(o_i|s)]$$

- The variable $v_i(s)$ represents the maximum probability that the $i$-th state is $s$, given that we have seen $O_1^i$.
- At each step, we record backpointers showing which previous state led to the maximum probability.

# Recall: Dynamic Programming


- Dynamic programming algorithms
  - solve large problems by compounding answers from smaller sub-problems
  - record sub-problem solutions for repeated use

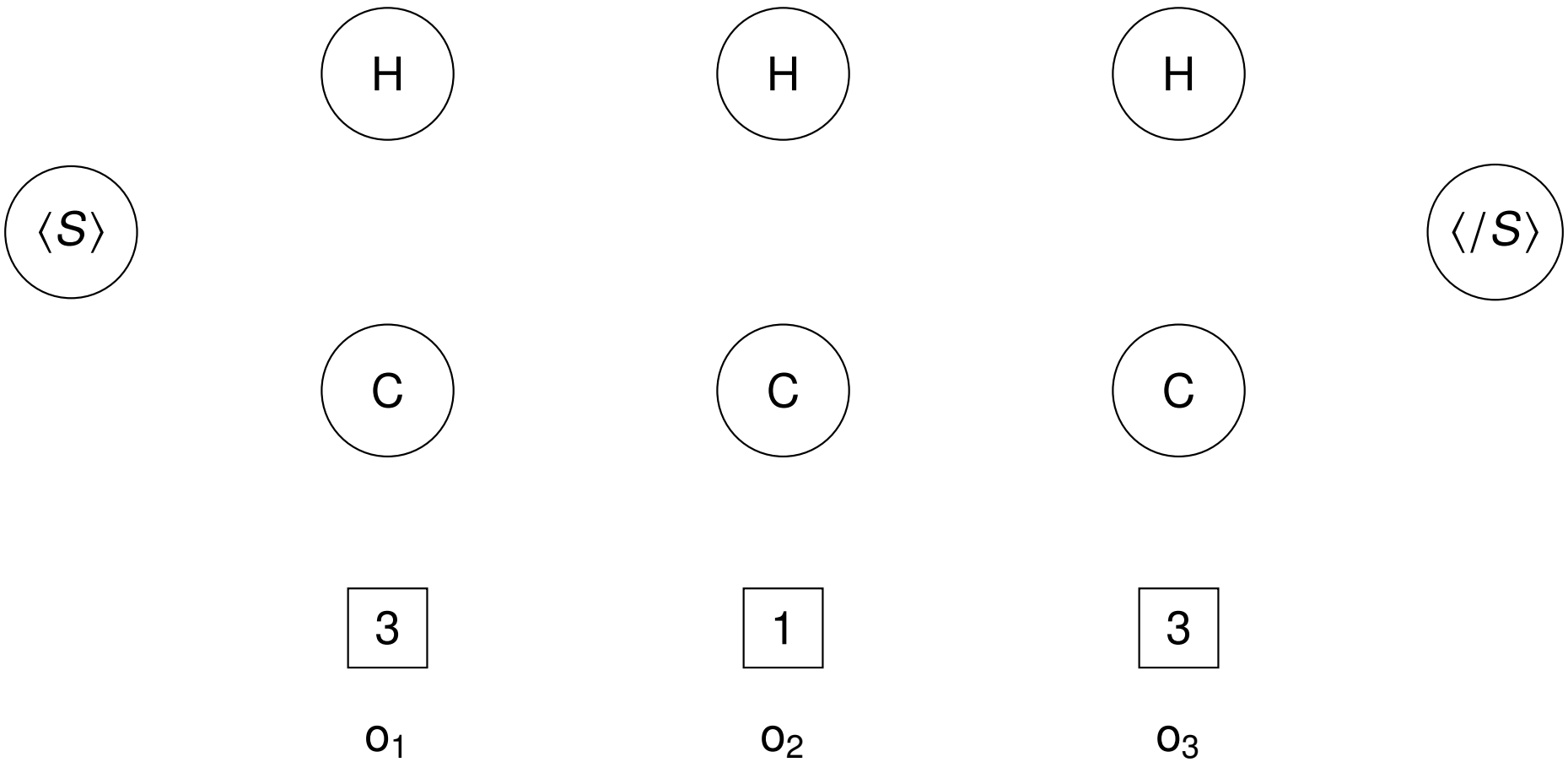
# Recall: Dynamic Programming


- Dynamic programming algorithms
  - solve large problems by compounding answers from smaller sub-problems
  - record sub-problem solutions for repeated use
- They are used for complex problems that
  - can be described recursively
  - require the same calculations over and over again

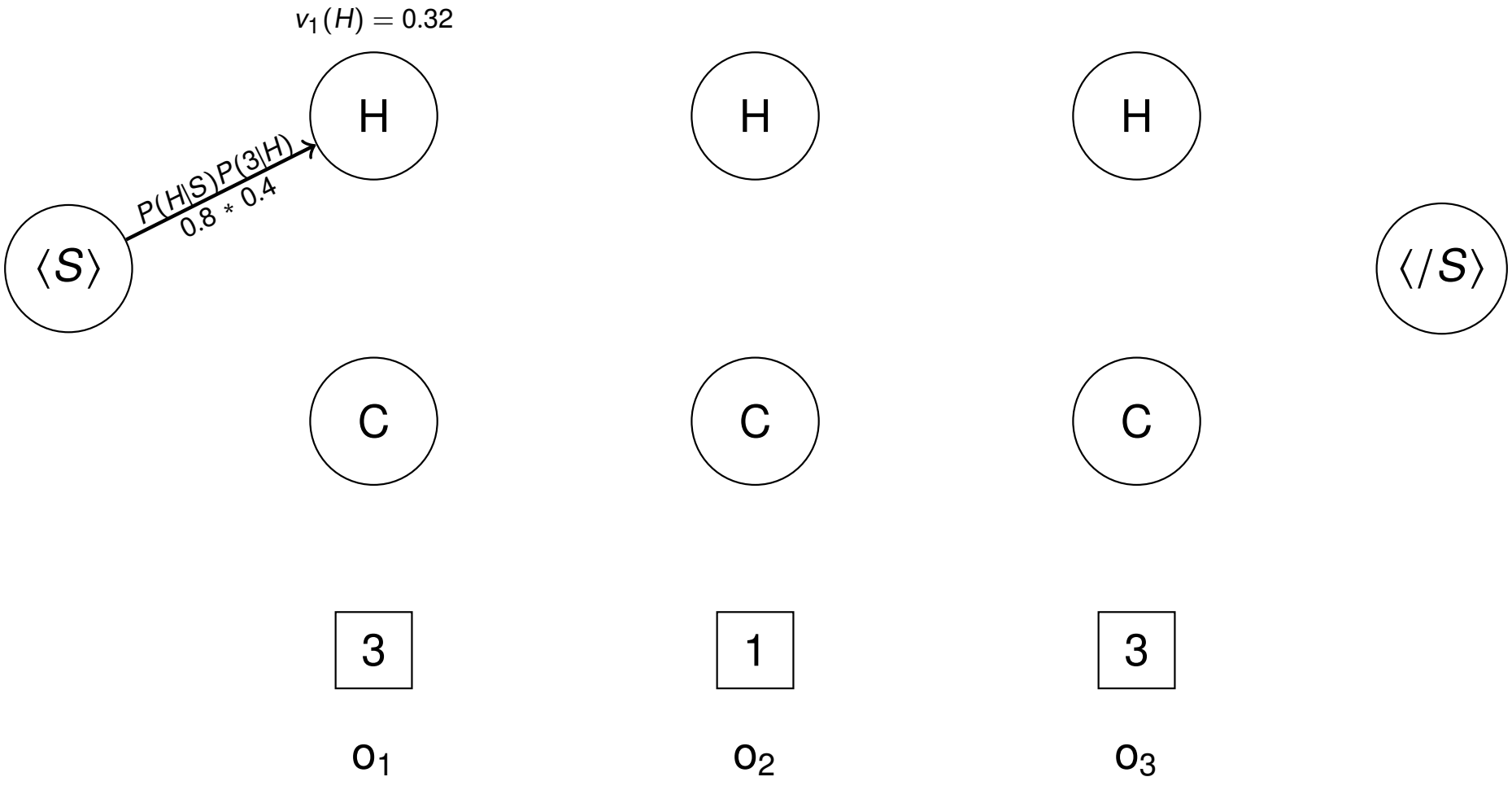
# Recall: Dynamic Programming


- Dynamic programming algorithms
  - solve large problems by compounding answers from smaller sub-problems
  - record sub-problem solutions for repeated use
- They are used for complex problems that
  - can be described recursively
  - require the same calculations over and over again
- Examples:
  - Dijkstra's shortest path
  - minimum edit distance
  - longest common subsequence
  - Viterbi decoding

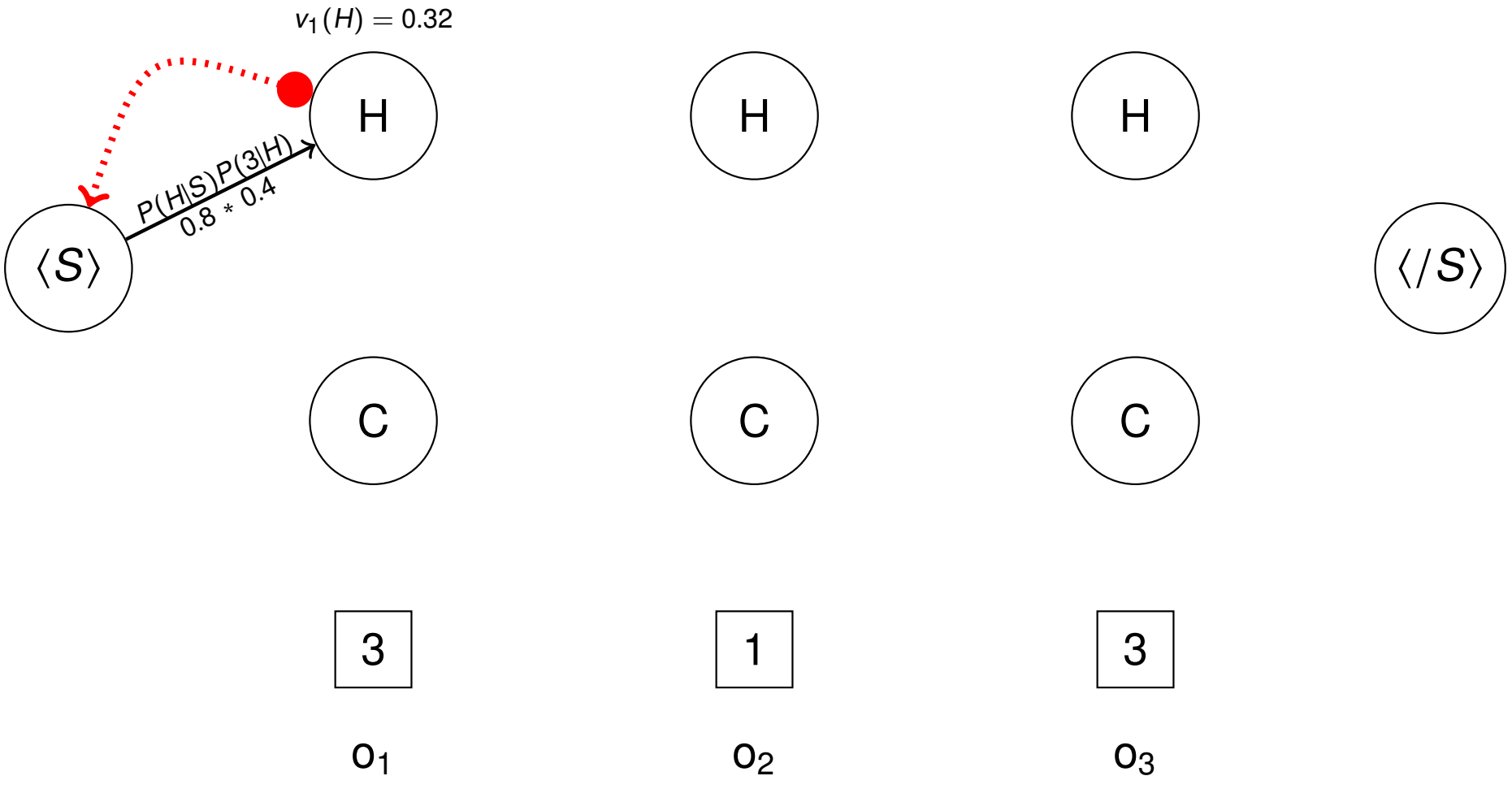
# Recall: An Example of the Viterbi Algorithmn



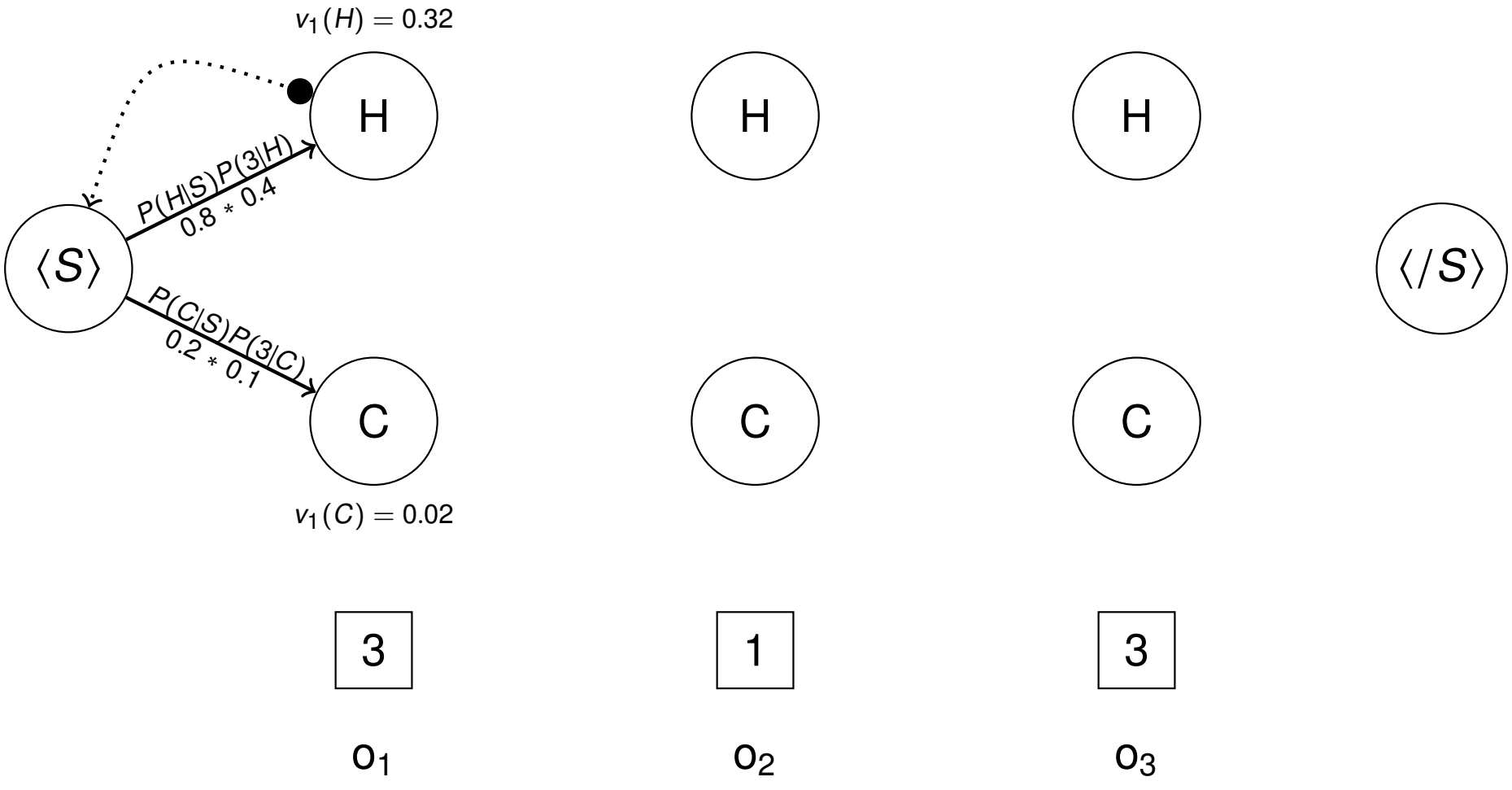
# Recall: An Example of the Viterbi Algorithmn



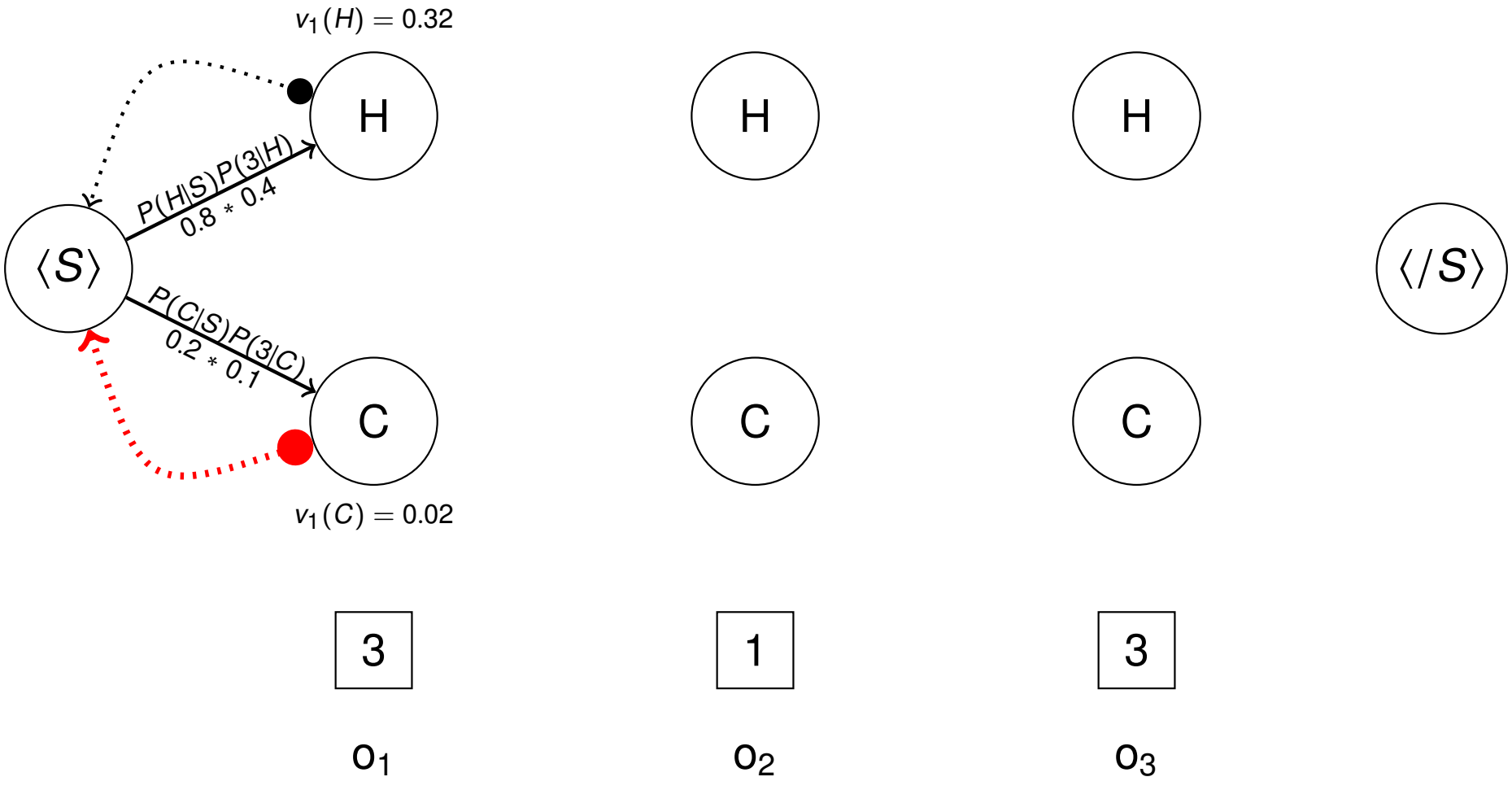
# Recall: An Example of the Viterbi Algorithmn



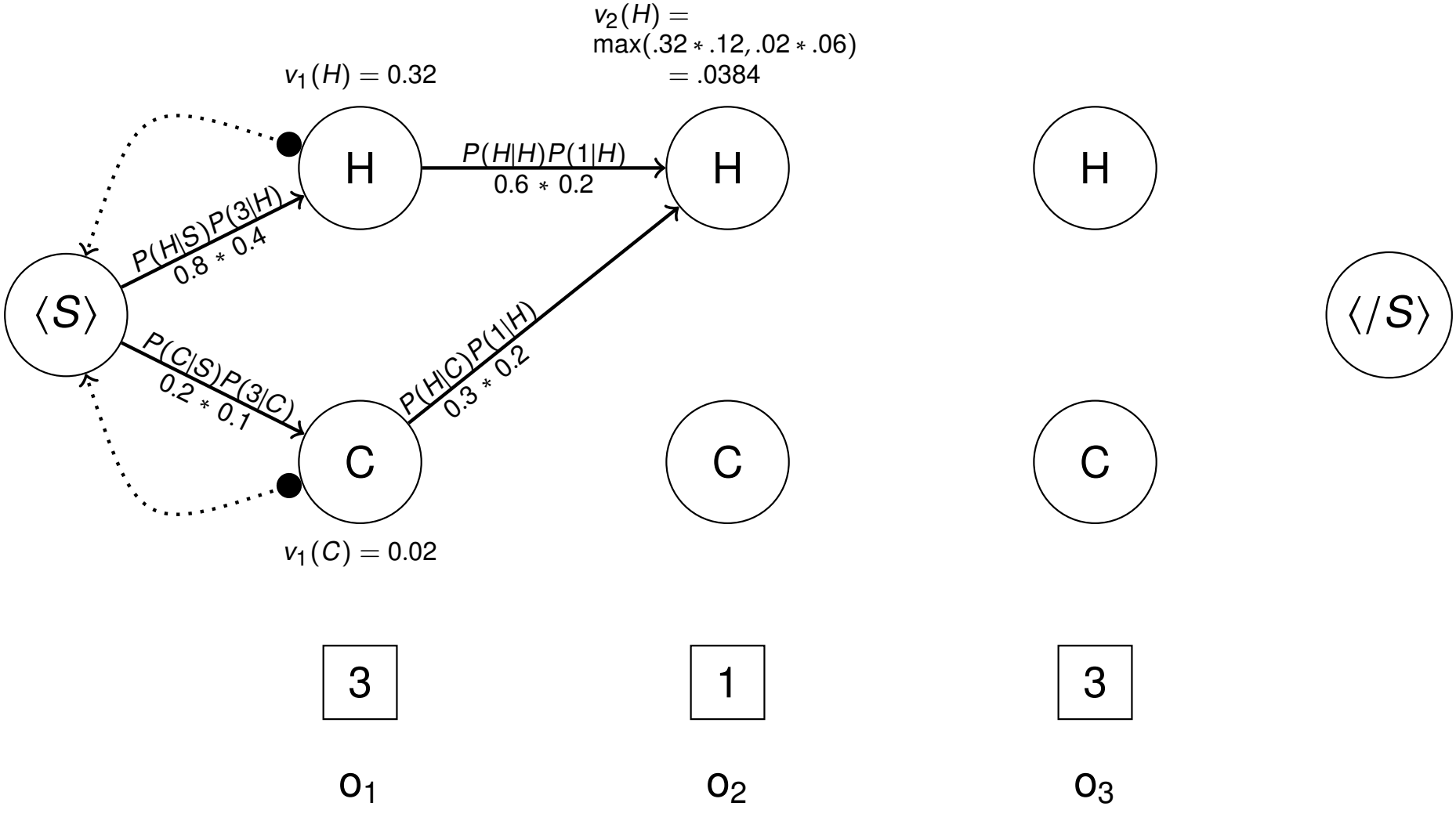
# Recall: An Example of the Viterbi Algorithmn



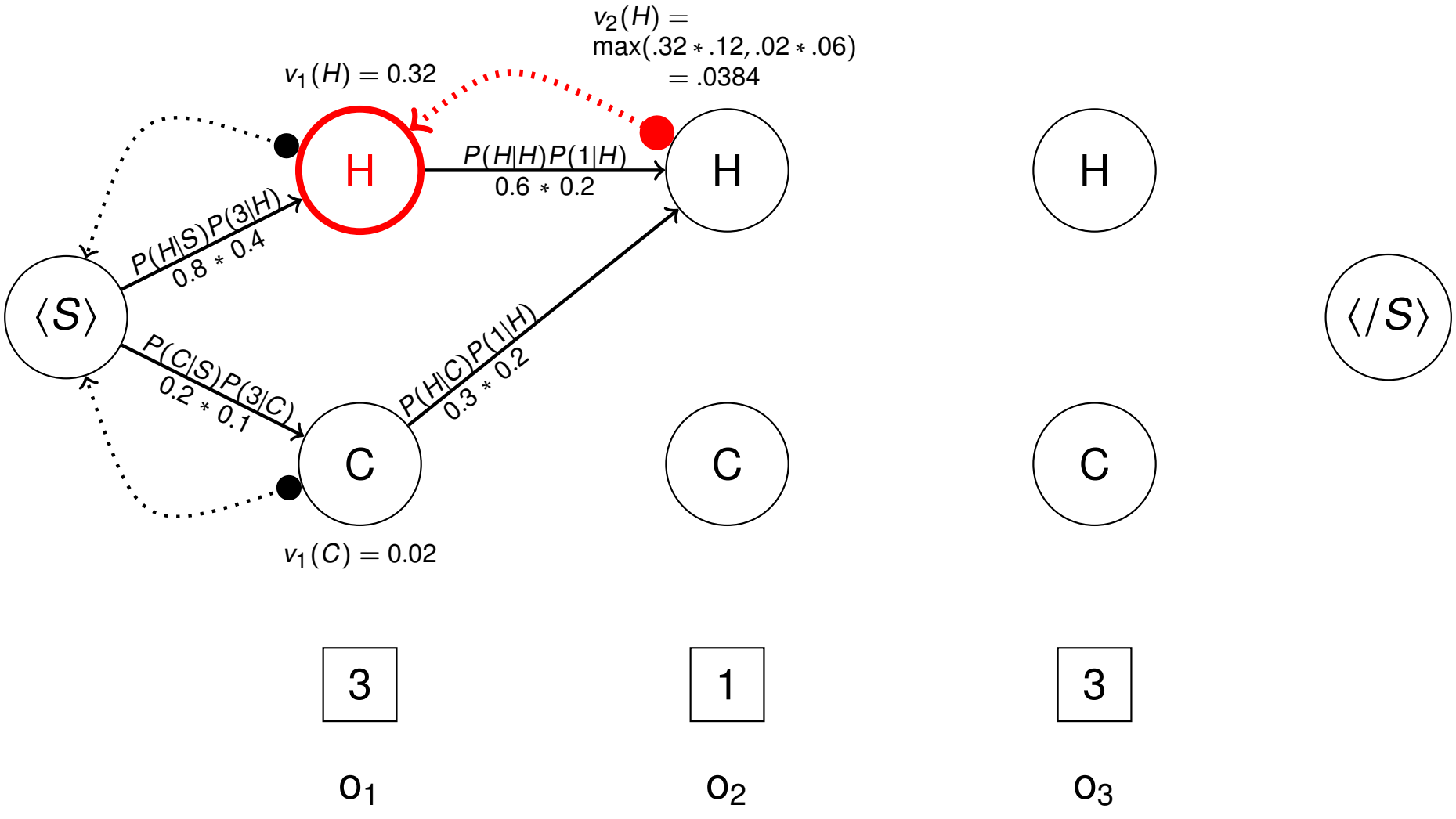
# Recall: An Example of the Viterbi Algorithmn



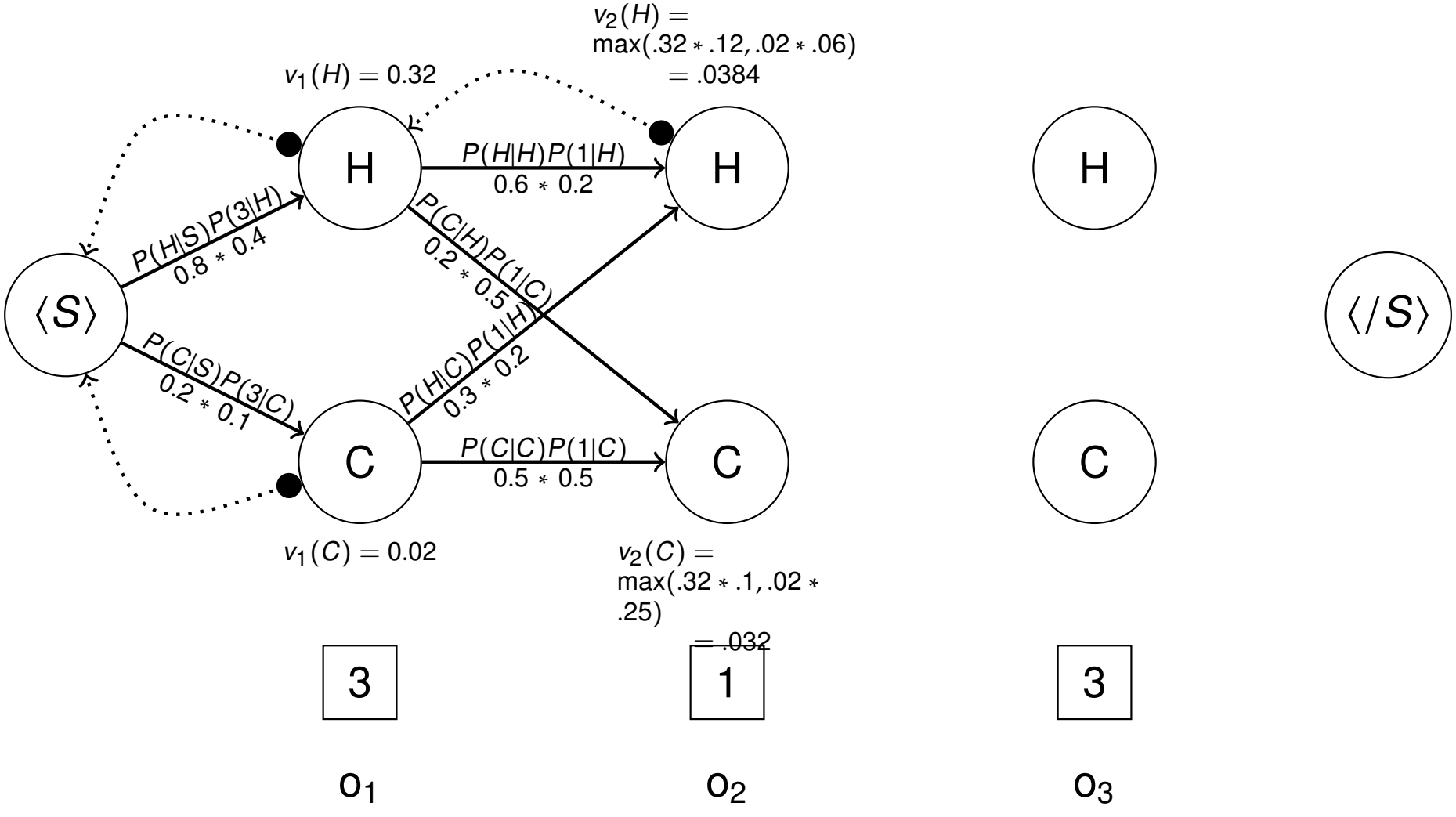
# Recall: An Example of the Viterbi Algorithmn



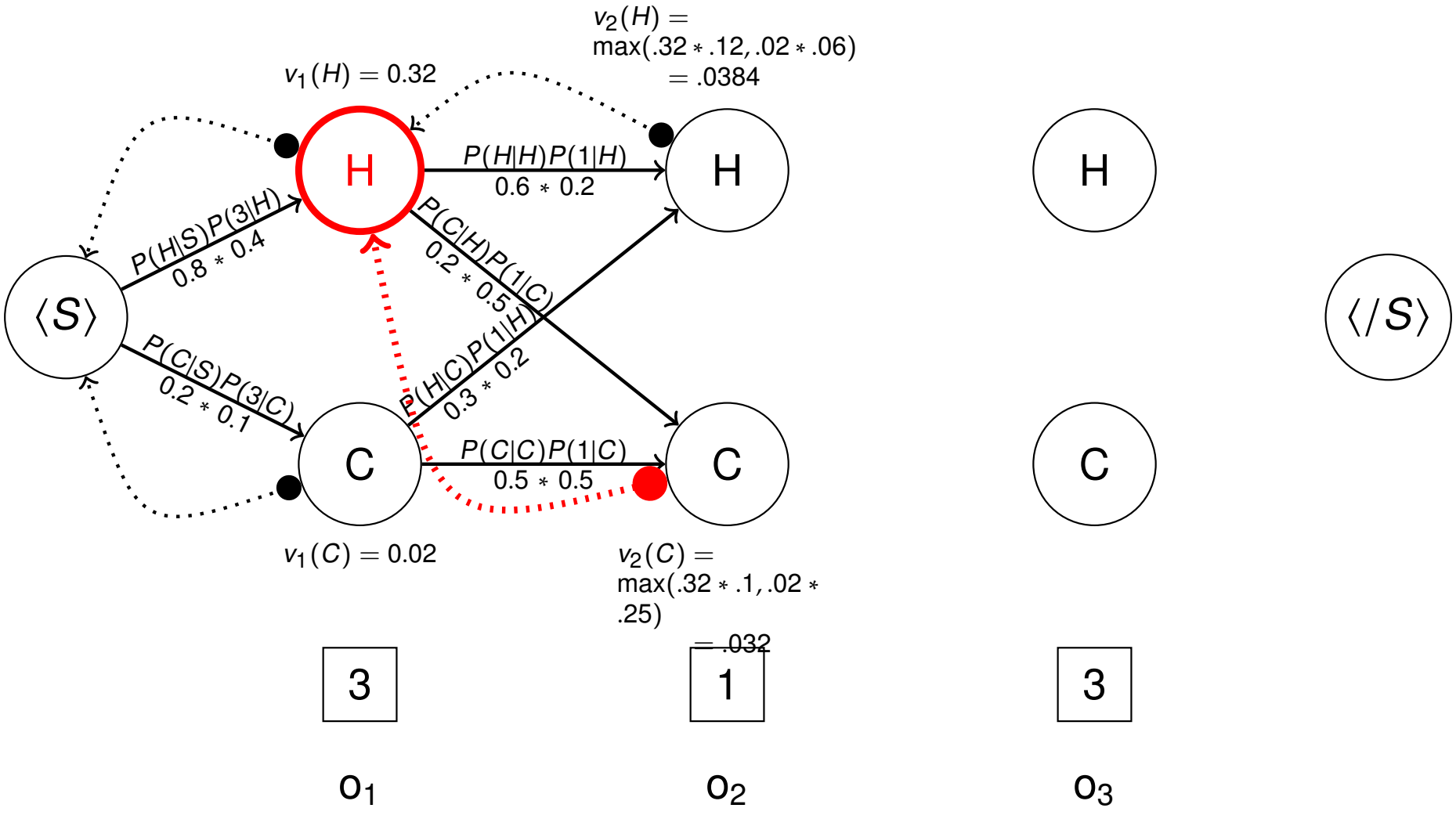
# Recall: An Example of the Viterbi Algorithmn



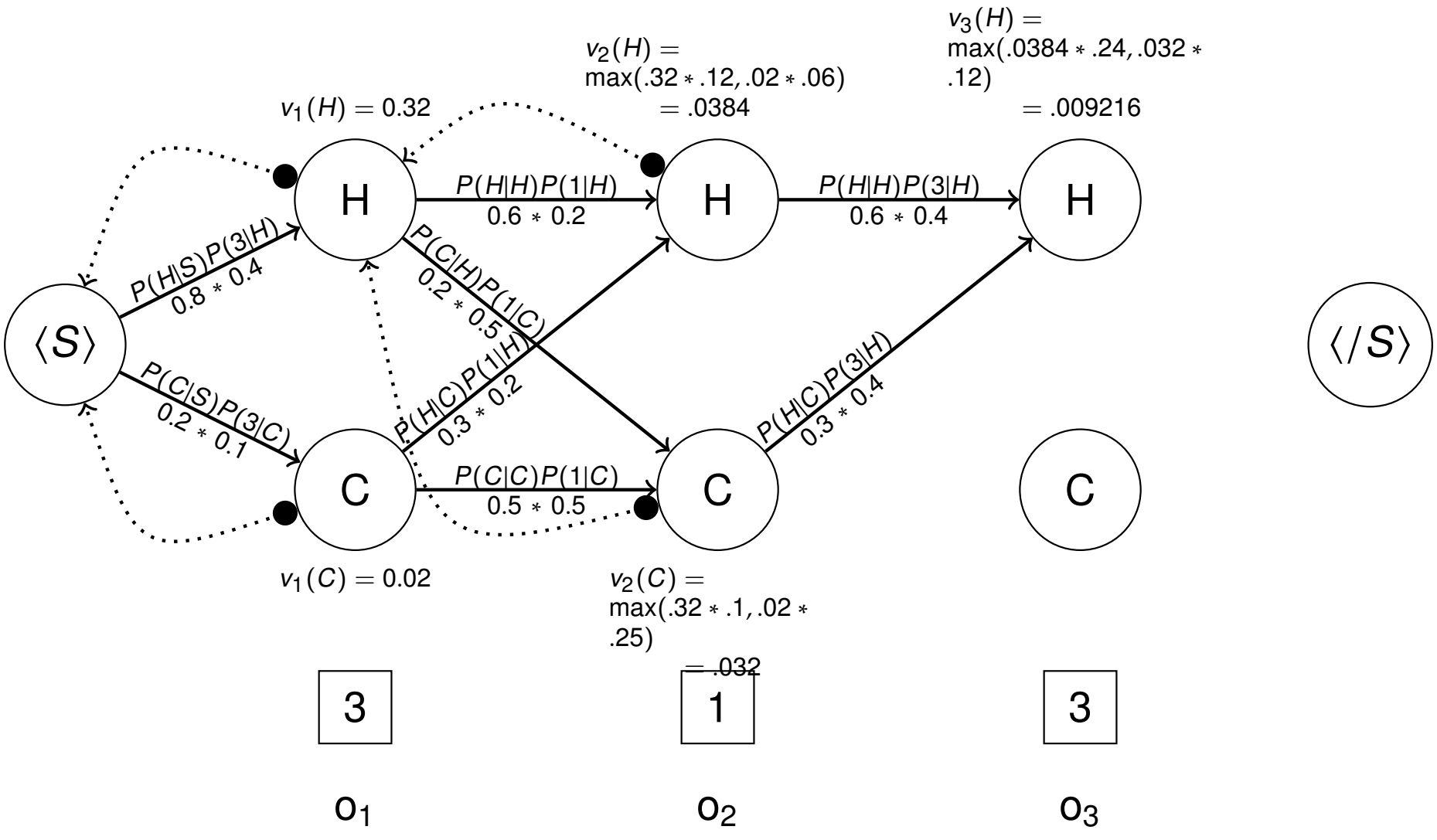
# Recall: An Example of the Viterbi Algorithmn



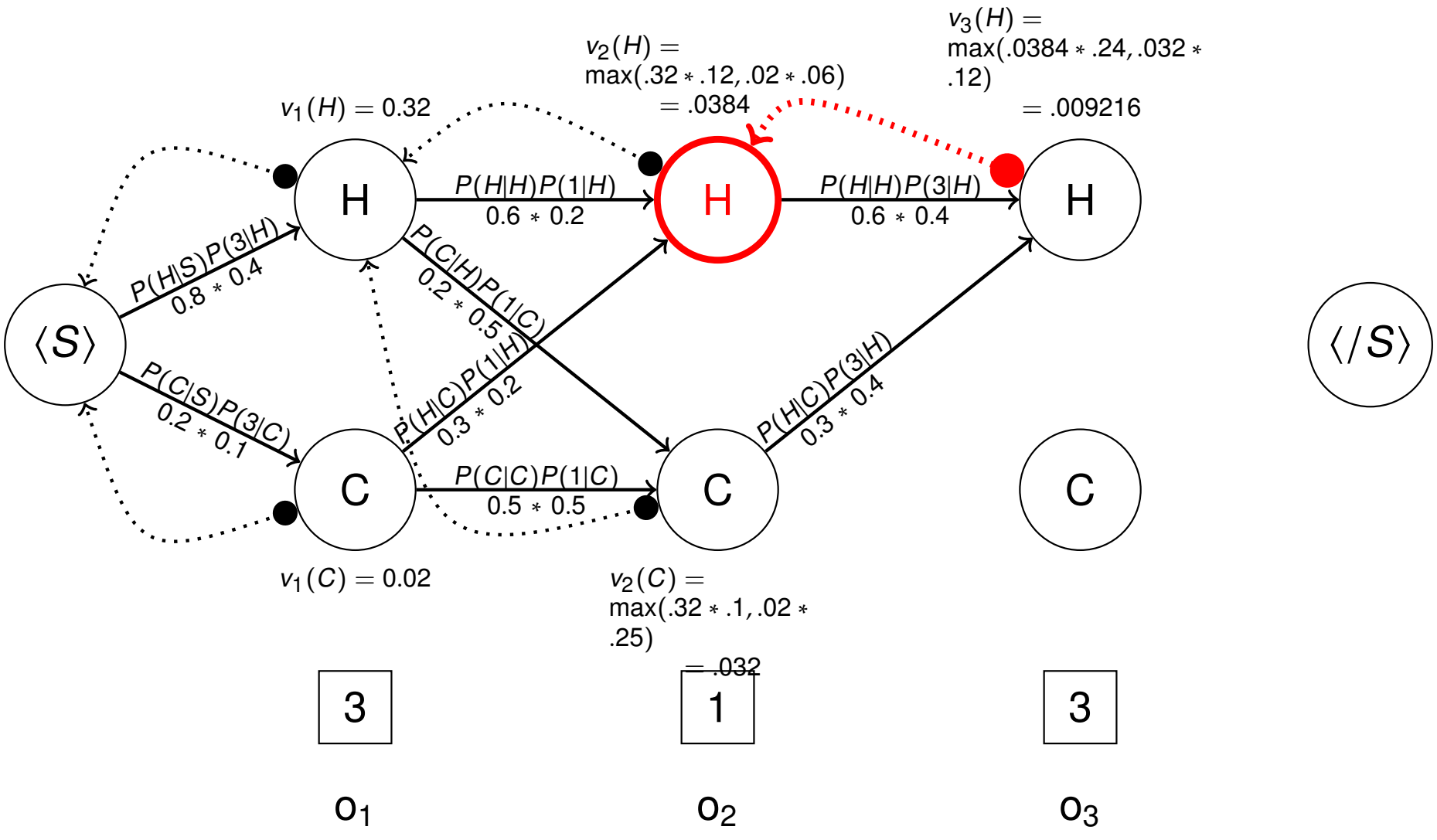
# Recall: An Example of the Viterbi Algorithmn



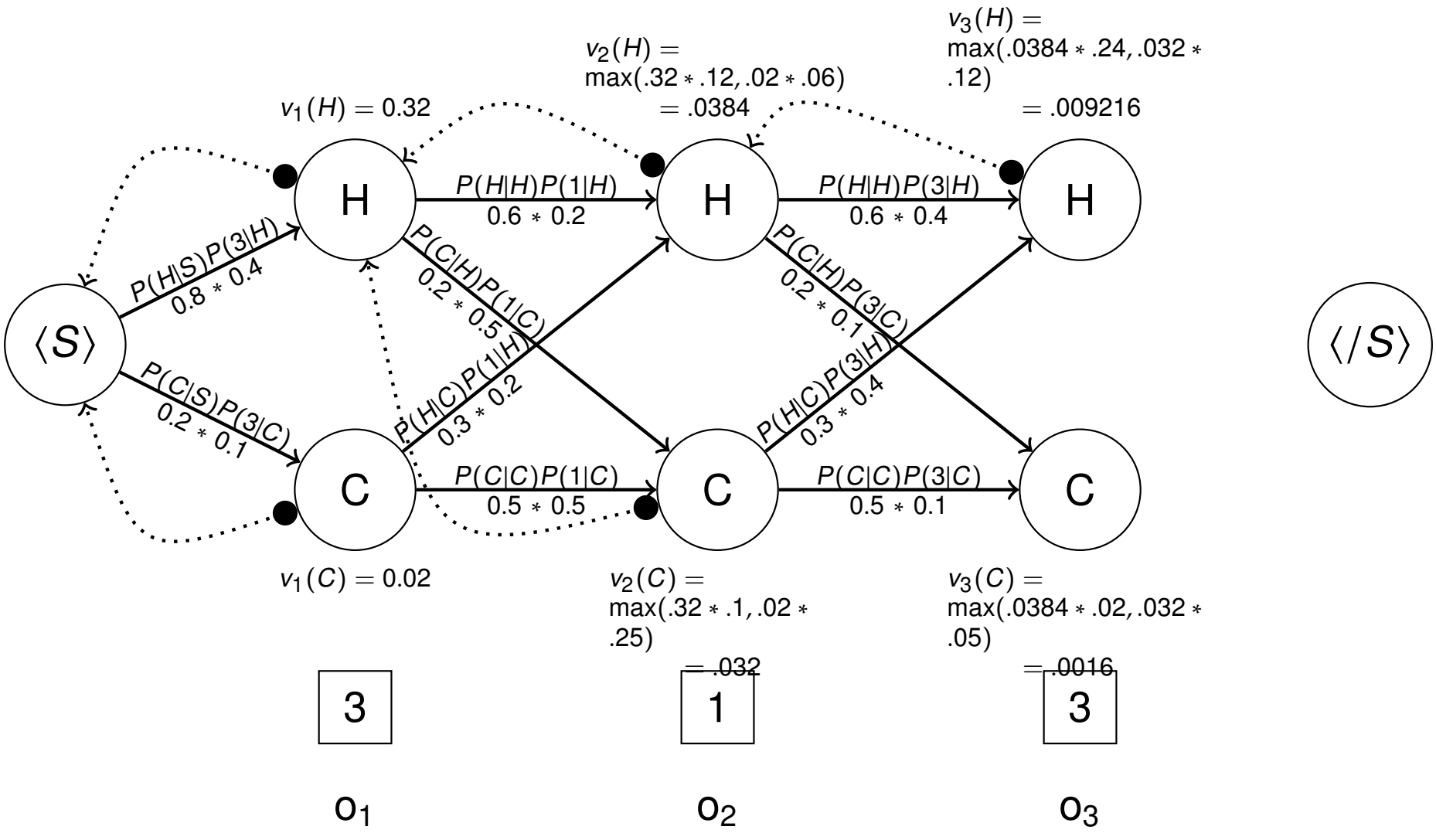
# Recall: An Example of the Viterbi Algorithmn



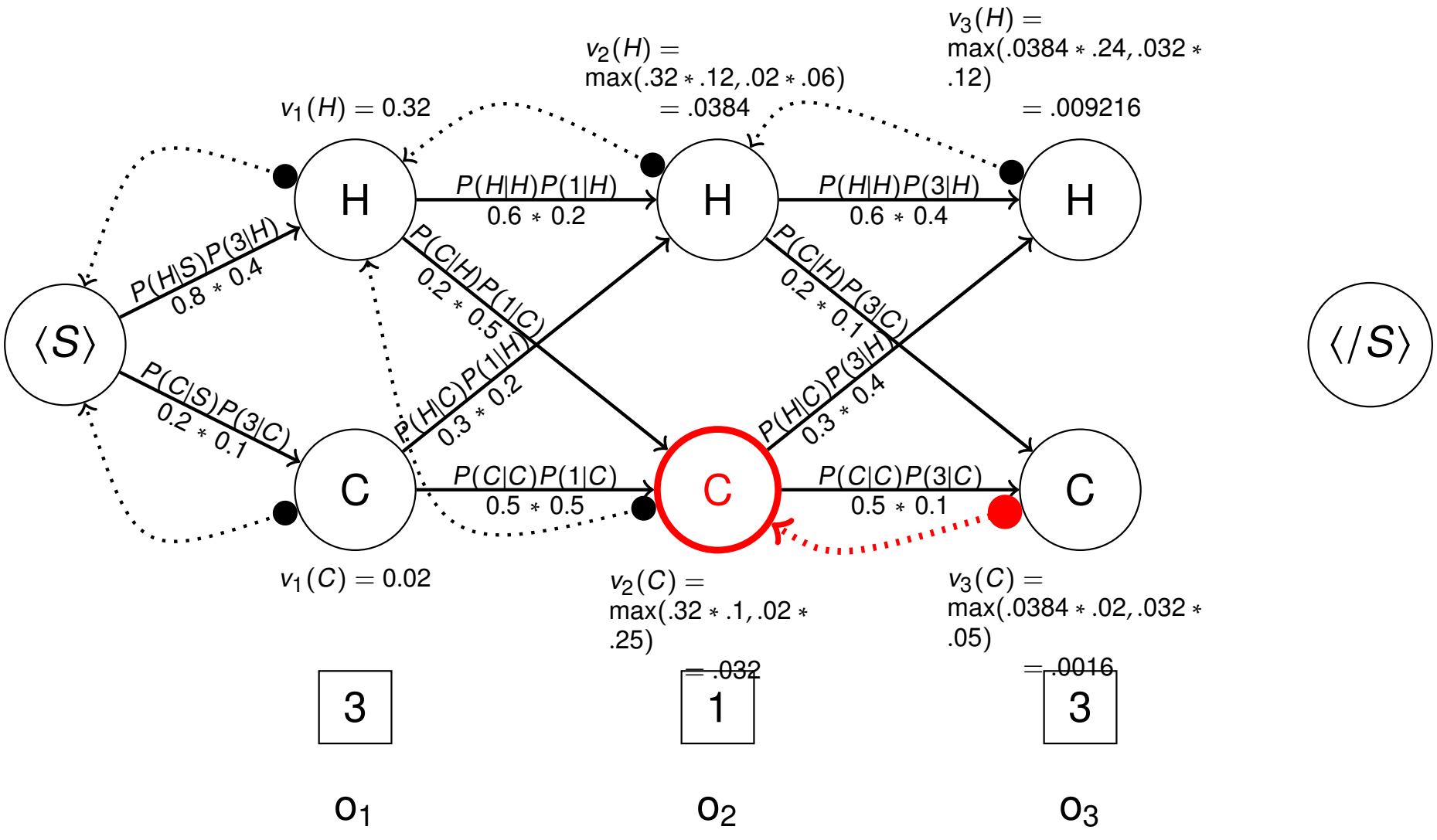
# Recall: An Example of the Viterbi Algorithmn



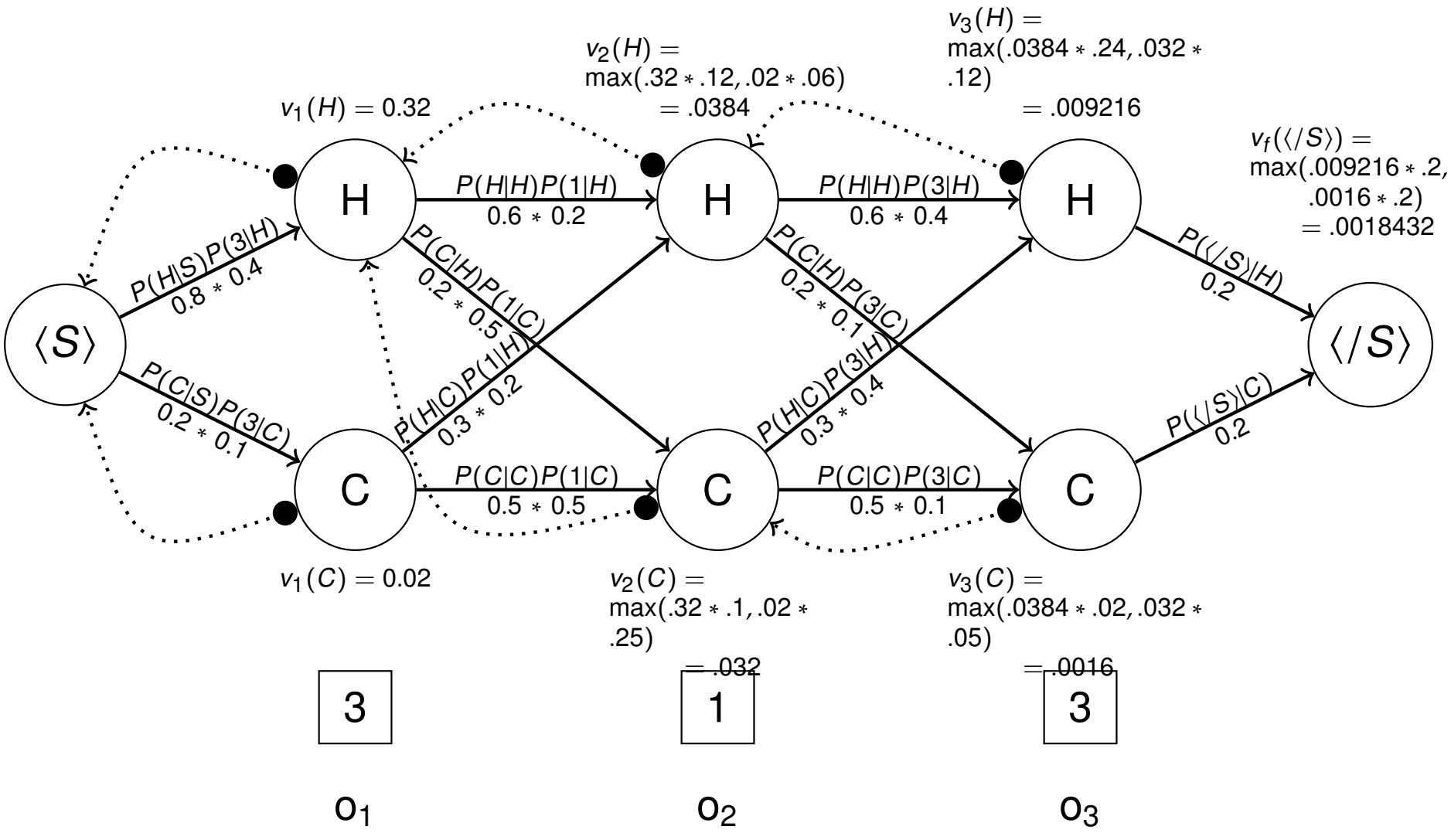
# Recall: An Example of the Viterbi Algorithmn



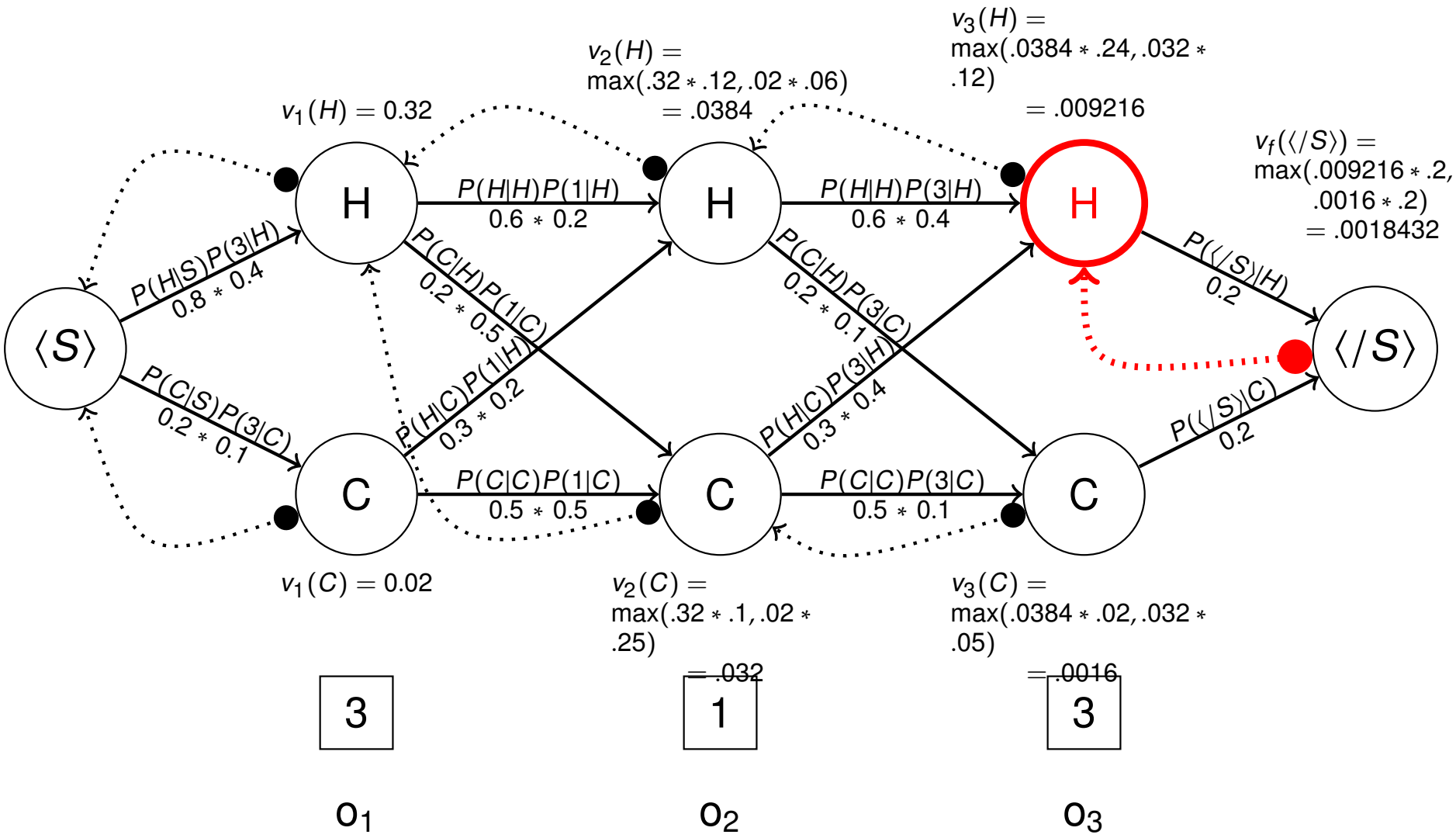
# Recall: An Example of the Viterbi Algorithmn



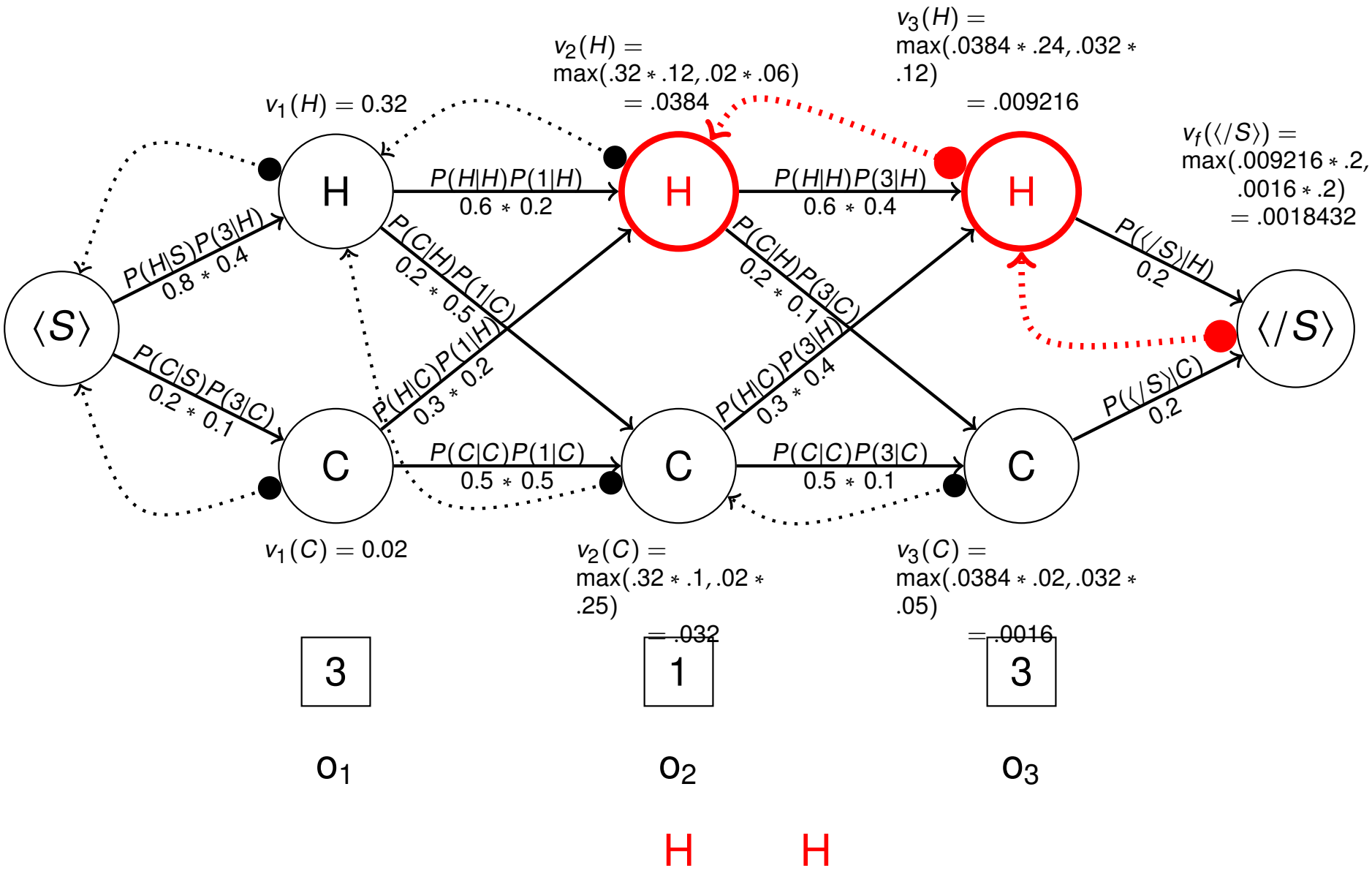
# Recall: An Example of the Viterbi Algorithmn



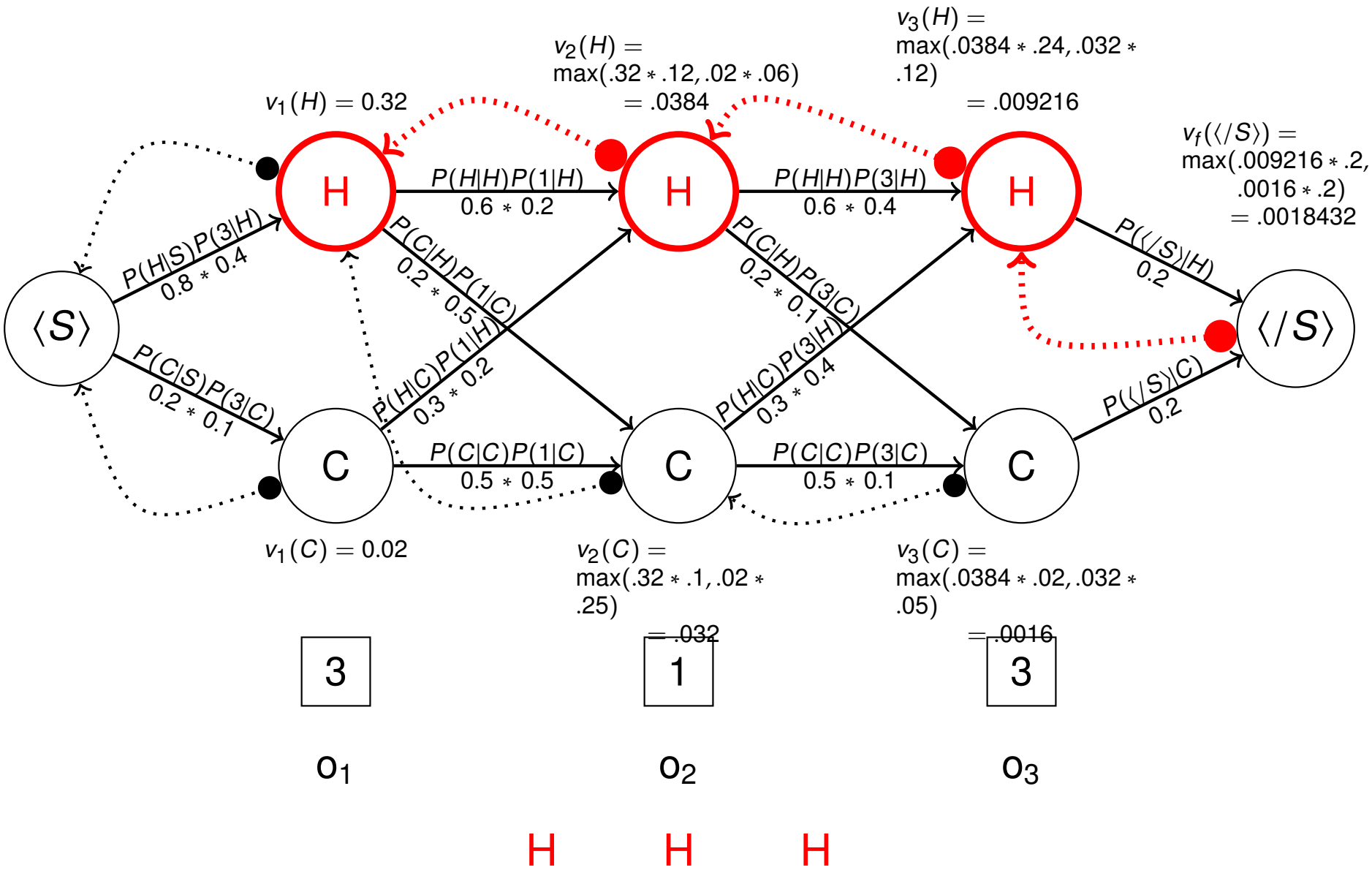
# Recall: An Example of the Viterbi Algorithmn



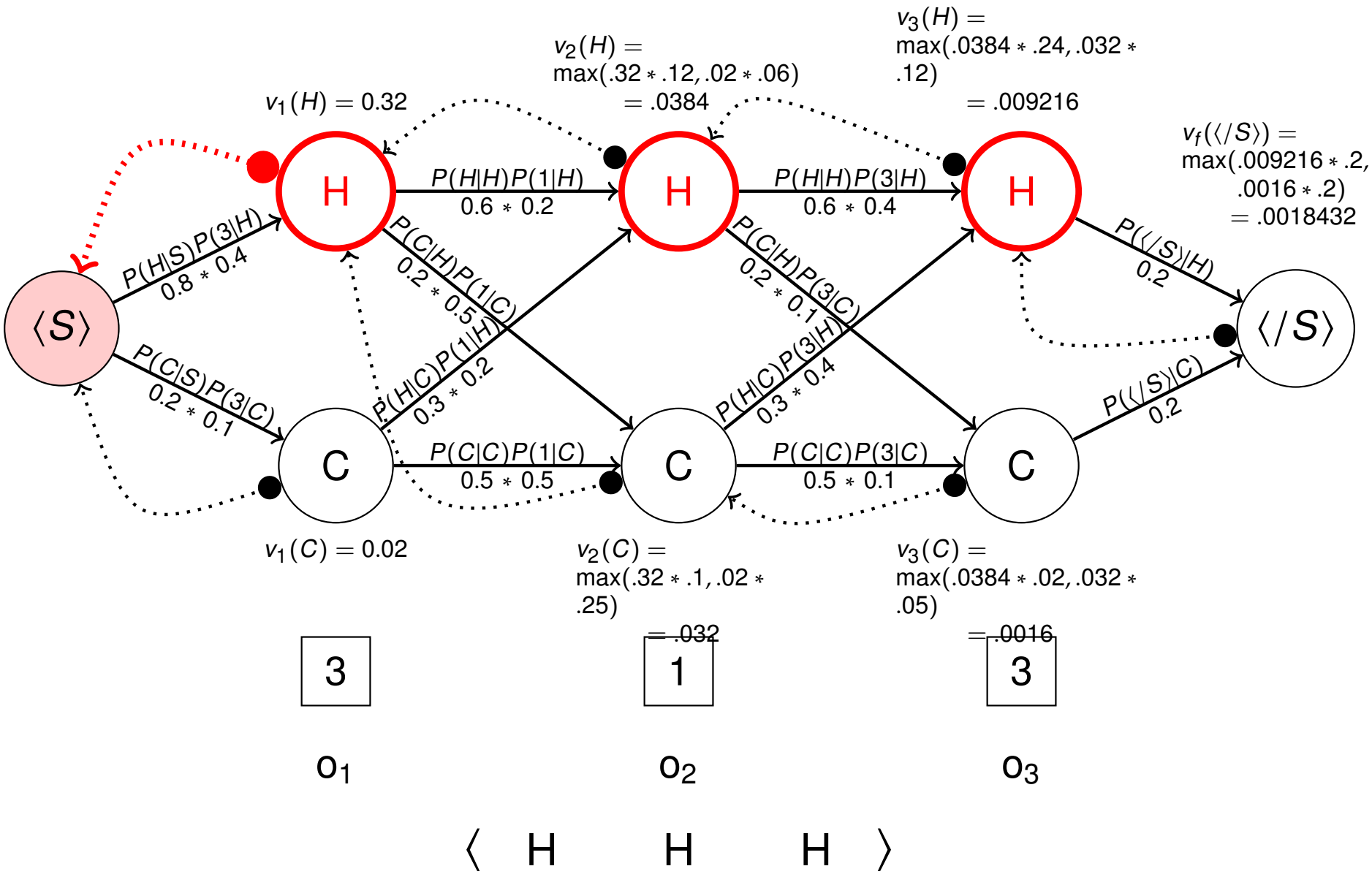
# Recall: An Example of the Viterbi Algorithmn



# Recall: An Example of the Viterbi Algorithmn



# Recall: An Example of the Viterbi Algorithmn



# Recall: Using HMMs

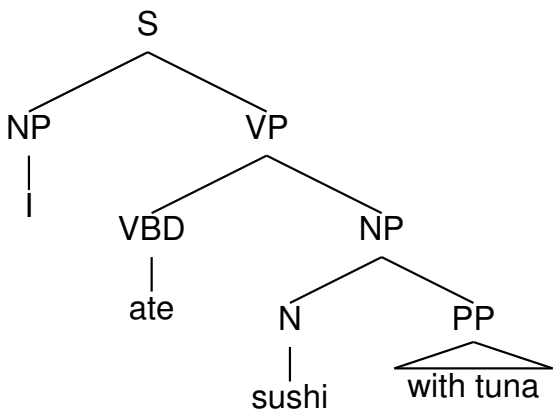

The HMM models the process of generating the labelled sequence. We can use this model for a number of tasks:

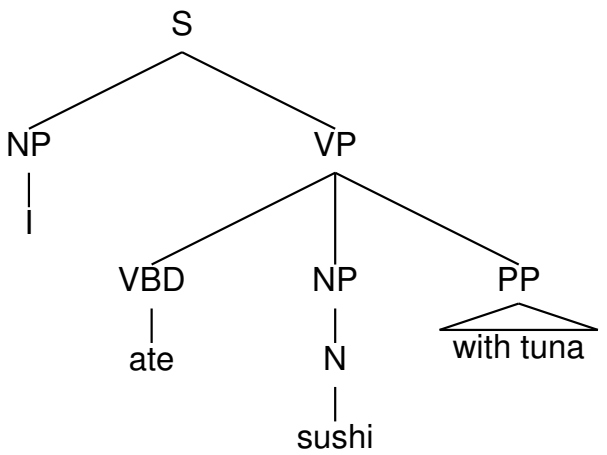
- $P(S, O)$ given $S$ and $O$
- $P(O)$ given $O$
- $S$ that maximizes $P(S|O)$ given $O$
- $P(s_x|O)$ given $O$
- We learn model parameters from a set of observations.

# Moving Onwards

Determining

- which string is most likely:
  - *How to recognize speech* vs. *How to wreck a nice beach*
- which tag sequence is most likely for *flies like flowers*:
  - **NNS VB NNS** vs. **VBZ P NNS**
- which syntactic structure is most likely:

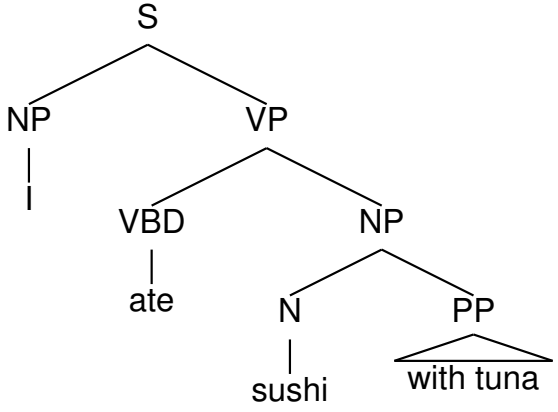


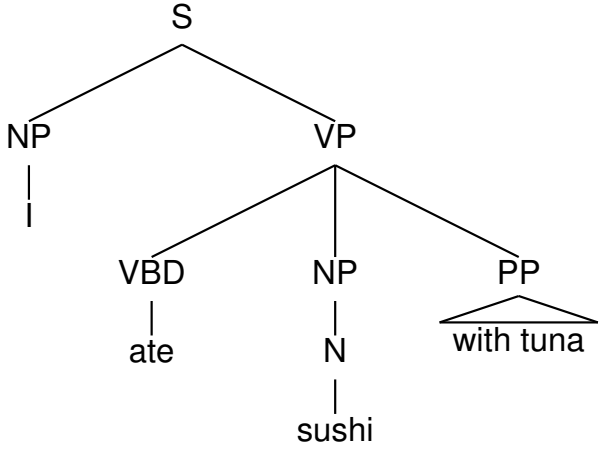



# Moving Onwards

Determining

- which string is most likely: ✓
  - *How to recognize speech* vs. *How to wreck a nice beach*
- which tag sequence is most likely for *flies like flowers*:
  - **NNS VB NNS** vs. **VBZ P NNS**
- which syntactic structure is most likely:

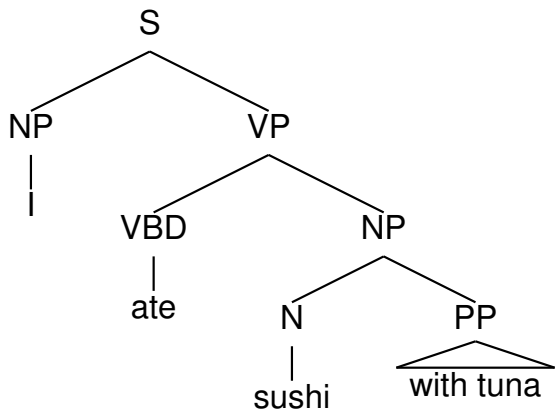


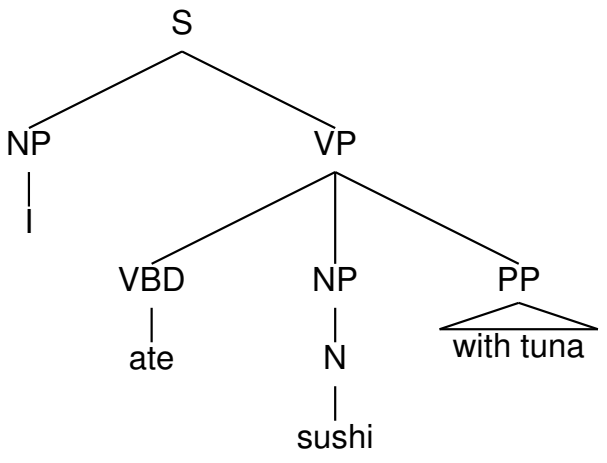



# Moving Onwards

Determining

- which string is most likely: ✓
  - *How to recognize speech* vs. *How to wreck a nice beach*
- which tag sequence is most likely for *flies like flowers*: ✓
  - **NNS VB NNS** vs. **VBZ P NNS**
- which syntactic structure is most likely:

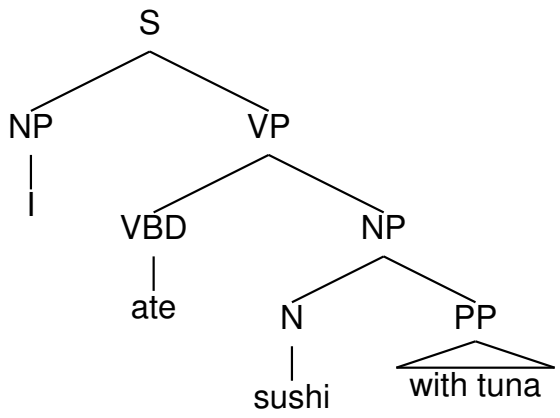


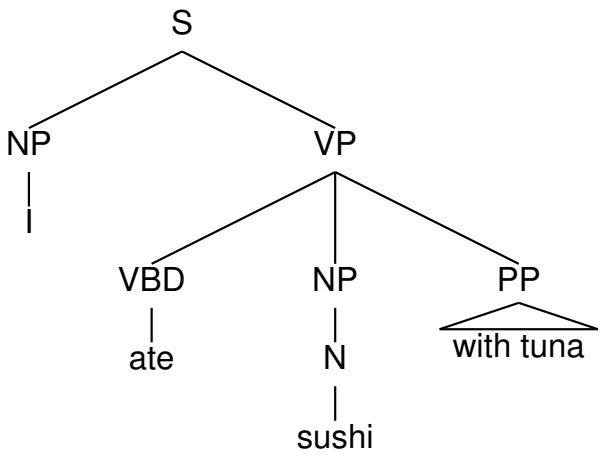



# Moving Onwards

Determining

- which string is most likely: ✓
  - *How to recognize speech* vs. *How to wreck a nice beach*
- which tag sequence is most likely for *flies like flowers*: ✓
  - **NNS VB NNS** vs. **VBZ P NNS**
- which syntactic structure is most likely:



# From Linear Order to Hierarchical Structure


- The models we have looked at so far:
  - *n*-gram models (Markov chains).
    - Purely linear (sequential) and surface-oriented.
  - sequence labeling: HMMs.
    - Adds one layer of abstraction: PoS as hidden variables.
    - Still only sequential in nature.

# From Linear Order to Hierarchical Structure


- The models we have looked at so far:
  - *n*-gram models (Markov chains).
    - Purely linear (sequential) and surface-oriented.
  - sequence labeling: HMMs.
    - Adds one layer of abstraction: PoS as hidden variables.
    - Still only sequential in nature.
- Formal grammar adds hierarchical structure.
  - In NLP, being a sub-discipline of AI, we want our programs to *'understand'* natural language (on some level).
  - Finding the grammatical structure of sentences is an important step towards 'understanding'.
  - Shift focus from *sequences* to *grammatical structures*.

# Why We Need Structure (1/3)


**Constituency**

- Words tends to lump together into groups that behave like single units: we call them *constituents*.
- *Constituency tests* give evidence for constituent structure:
  - interchangeable in similar syntactic environments.
  - can be co-ordinated
  - can be moved within a sentence as a unit

# Why We Need Structure (1/3)


## Constituency

- Words tends to lump together into groups that behave like single units: we call them *constituents*.
- *Constituency tests* give evidence for constituent structure:
  - interchangeable in similar syntactic environments.
  - can be co-ordinated
  - can be moved within a sentence as a unit

(4) Kim read [a very interesting book about grammar]$_{NP}$.
Kim read [it]$_{NP}$.

(5) Kim [read a book]$_{VP}$, [gave it to Sandy]$_{VP}$, and [left]$_{VP}$.

(6) [Read the book]$_{VP}$ I really meant to this week.

Examples from *Linguistic Fundamentals for NLP: 100 Essentials from Morphology and Syntax*. Bender (2013)

# Why We Need Structure (2/3)


**Constituency**

- Constituents are theory-dependent, and are not absolute or language-independent.

# Why We Need Structure (2/3)


**Constituency**

- Constituents are theory-dependent, and are not absolute or language-independent.
- A constituent usually has a *head* element, and is often named according to the type of its head:
  - A noun phrase (NP) has a nominal (noun-type) head:

    (9) [ a very interesting book about grammar ]$_{\text{NP}}$

  - A verb phrase (VP) has a verbal head:

    (10) [ gives books to students ]$_{\text{VP}}$

# Why We Need Structure (3/3)


**Grammatical functions**

- Terms such as *subject* and *object* describe the grammatical function of a constituent in a sentence.

# Why We Need Structure (3/3)


**Grammatical functions**

- Terms such as *subject* and *object* describe the grammatical function of a constituent in a sentence.
- *Agreement* establishes a symmetric relationship between grammatical features.

  The *decision* of the Nobel committee member*s* surprise*s* most of us.

- Why would a purely linear model have problems predicting this phenomenon?

# Why We Need Structure (3/3)


**Grammatical functions**

- Terms such as *subject* and *object* describe the grammatical function of a constituent in a sentence.
- *Agreement* establishes a symmetric relationship between grammatical features.

  The *decision* of the Nobel committee member*s* surprise*s* most of us.

- Why would a purely linear model have problems predicting this phenomenon?
- Verb agreement reflects the grammatical structure of the sentence, not just the sequential order of words.

# Grammars: A Tool to Aid Understanding


Formal grammars describe a language, giving us a way to:

- judge or predict well-formedness

  Kim was happy because _____ passed the exam.

  Kim was happy because _____ final grade was an A.

# Grammars: A Tool to Aid Understanding


Formal grammars describe a language, giving us a way to:

- judge or predict well-formedness

  Kim was happy because _____ passed the exam.

  Kim was happy because _____ final grade was an A.

- make explicit structural ambiguities

  Have her report on my desk by Friday!

  I like to eat sushi with {chopsticks|tuna}.

# Grammars: A Tool to Aid Understanding


Formal grammars describe a language, giving us a way to:

- judge or predict well-formedness

  Kim was happy because _____ passed the exam.

  Kim was happy because _____ final grade was an A.

- make explicit structural ambiguities

  Have her report on my desk by Friday!

  I like to eat sushi with {chopsticks|tuna}.

- derive abstract representations of meaning

  Kim gave Sandy a book.

  Kim gave a book to Sandy.

  Sandy was given a book by Kim.

# A Grossly Simplified Example


## The Grammar of Spanish

S → NP VP

VP → V NP

VP → VP PP

PP → P NP

NP → “nieve”

NP → “Juan”

NP → “Oslo”

V → “amó”

P → “en”

# A Grossly Simplified Example


## The Grammar of Spanish

S → NP VP

VP → V NP

VP → VP PP

PP → P NP

NP → “nieve”

NP → “Juan”

NP → “Oslo”

V → “amó”

P → “en”

# A Grossly Simplified Example


## The Grammar of Spanish

S → NP VP

VP → V NP

VP → VP PP

PP → P NP

NP → “nieve”

NP → “Juan”

NP → “Oslo”

V → “amó”

P → “en”

# A Grossly Simplified Example


## The Grammar of Spanish

S → NP VP

VP → V NP

VP → VP PP

PP → P NP

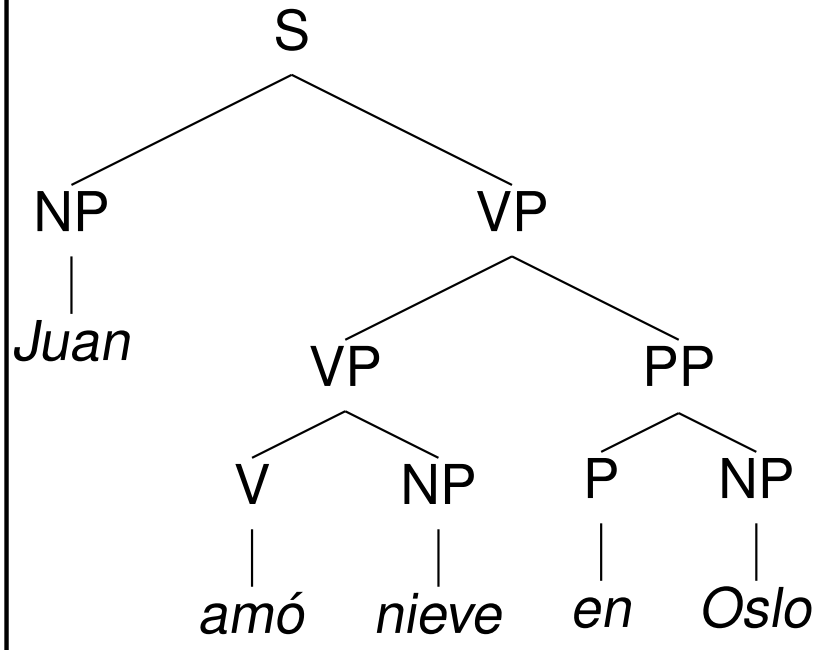
NP → “nieve”

NP → “Juan”

NP → “Oslo”

V → “amó”

P → “en”




*Juan amó nieve en Oslo*

# A Grossly Simplified Example


## The Grammar of Spanish

S → NP VP

VP → V NP

VP → VP PP

PP → P NP

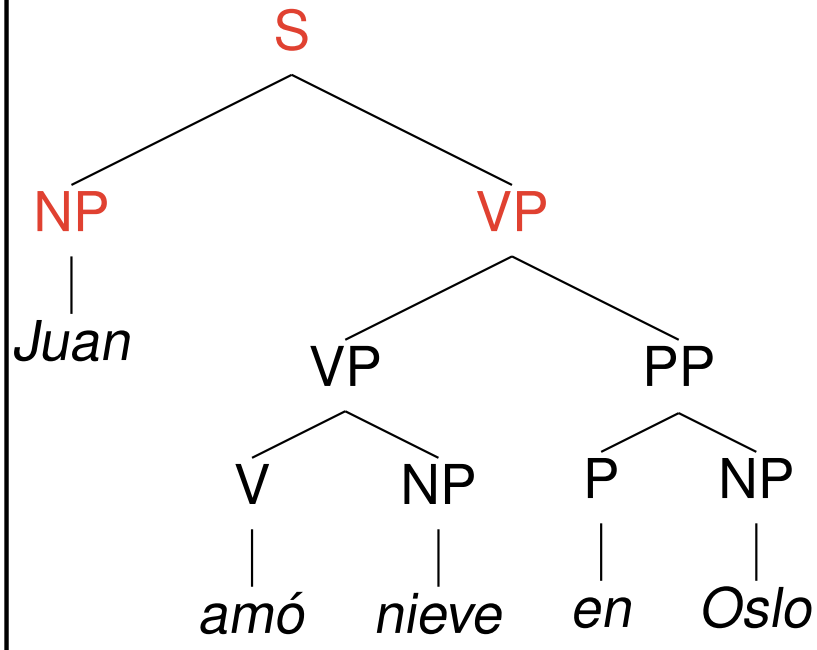
NP → “nieve”

NP → “Juan”

NP → “Oslo”

V → “amó”

P → “en”




*Juan amó nieve en Oslo*

# A Grossly Simplified Example


## The Grammar of Spanish

S → NP VP

VP → V NP

VP → VP PP

PP → P NP

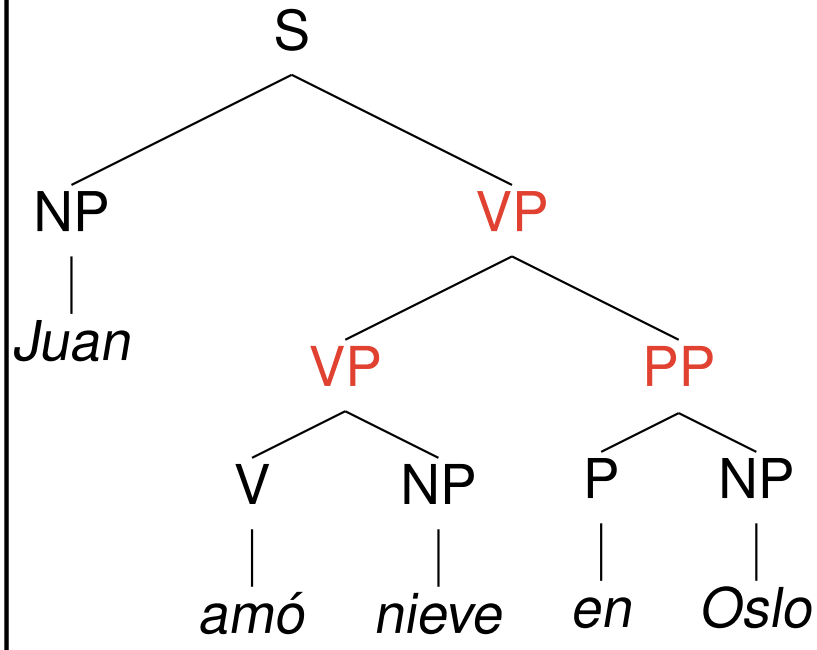
NP → “nieve”

NP → “Juan”

NP → “Oslo”

V → “amó”

P → “en”




*Juan amó nieve en Oslo*

# A Grossly Simplified Example


## The Grammar of Spanish

S → NP VP

VP → V NP

VP → VP PP

PP → P NP

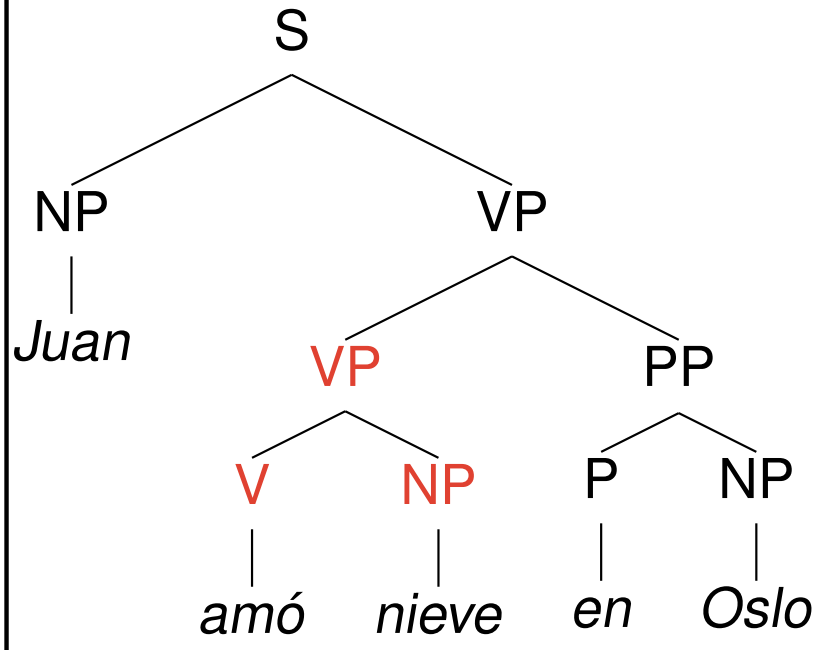
NP → “nieve”

NP → “Juan”

NP → “Oslo”

V → “amó”

P → “en”




*Juan amó nieve en Oslo*

# A Grossly Simplified Example


## The Grammar of Spanish

S → NP VP

VP → V NP

VP → VP PP

PP → P NP

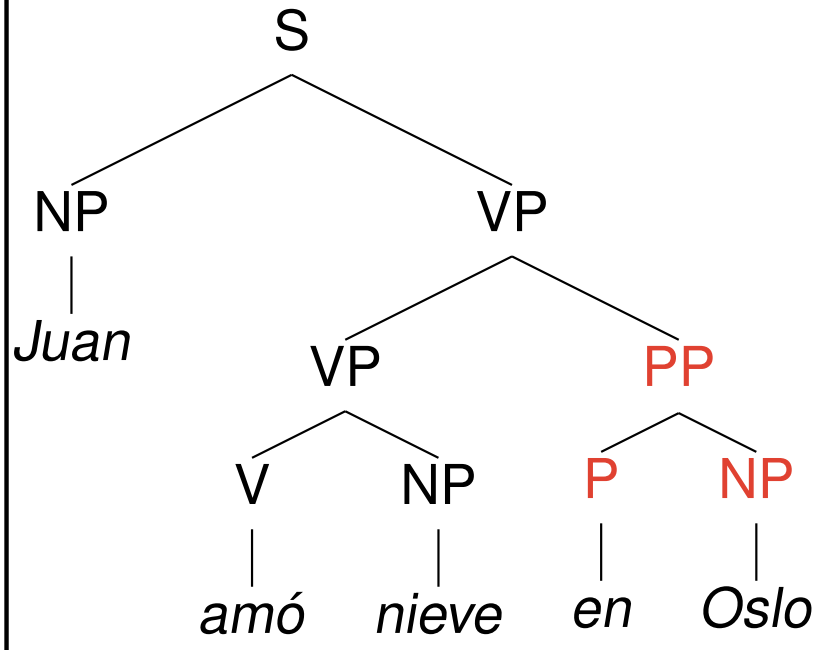
NP → “nieve”

NP → “Juan”

NP → “Oslo”

V → “amó”

P → “en”




*Juan amó nieve en Oslo*

# A Grossly Simplified Example


## The Grammar of Spanish

S → NP VP

VP → V NP

VP → VP PP

PP → P NP

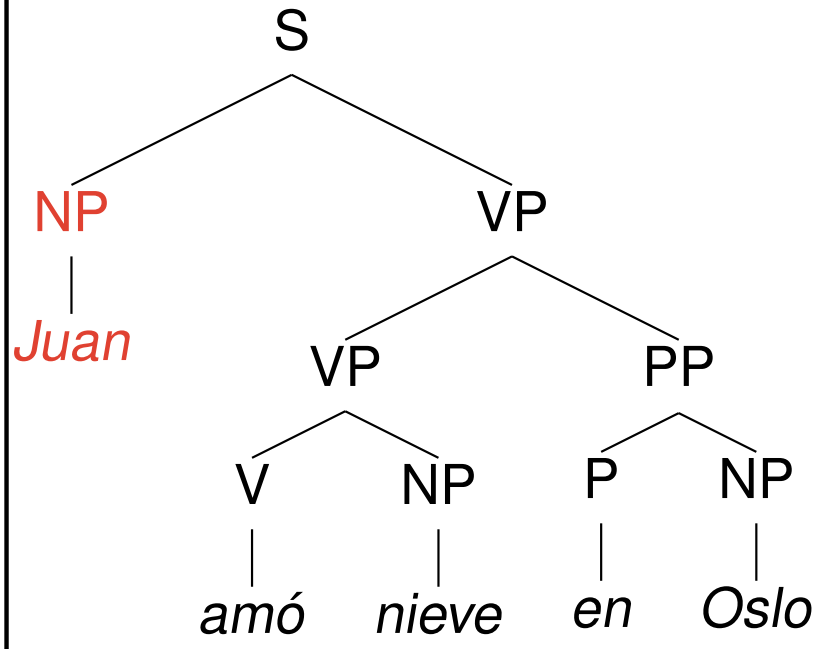
NP → “nieve”

NP → “Juan”

NP → “Oslo”

V → “amó”

P → “en”




*Juan amó nieve en Oslo*

# A Grossly Simplified Example


## The Grammar of Spanish

S → NP VP

VP → V NP

VP → VP PP

PP → P NP

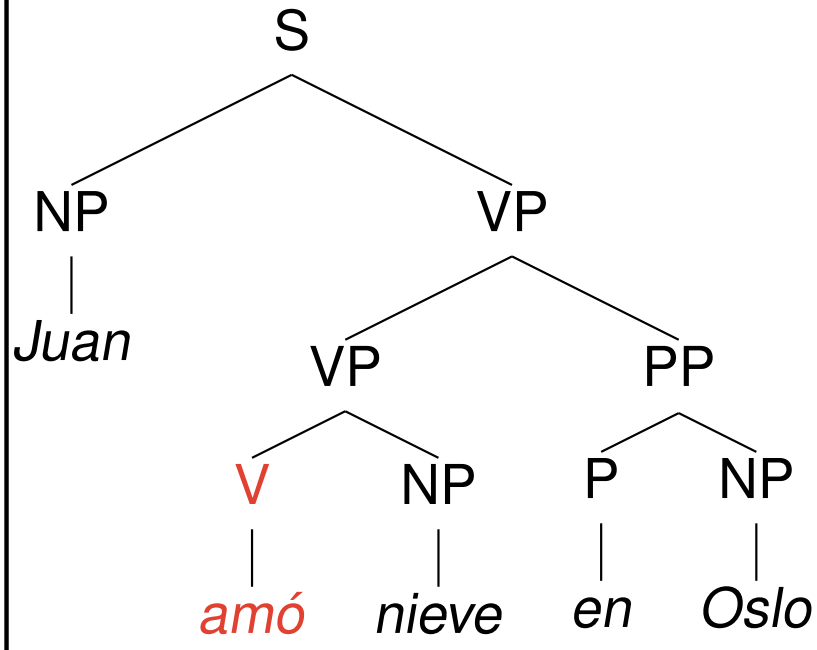
NP → “nieve”

NP → “Juan”

NP → “Oslo”

V → “amó”

P → “en”




*Juan amó nieve en Oslo*

# A Grossly Simplified Example


## The Grammar of Spanish

S → NP VP

VP → V NP

VP → VP PP

PP → P NP

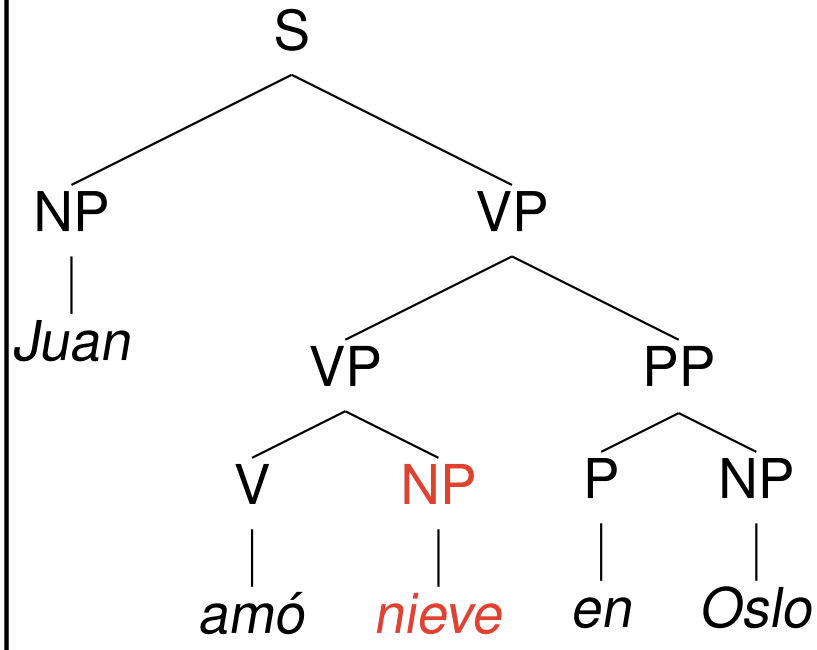
NP → “nieve”

NP → “Juan”

NP → “Oslo”

V → “amó”

P → “en”




*Juan amó nieve en Oslo*

# A Grossly Simplified Example


## The Grammar of Spanish

S → NP VP

VP → V NP

VP → VP PP

PP → P NP

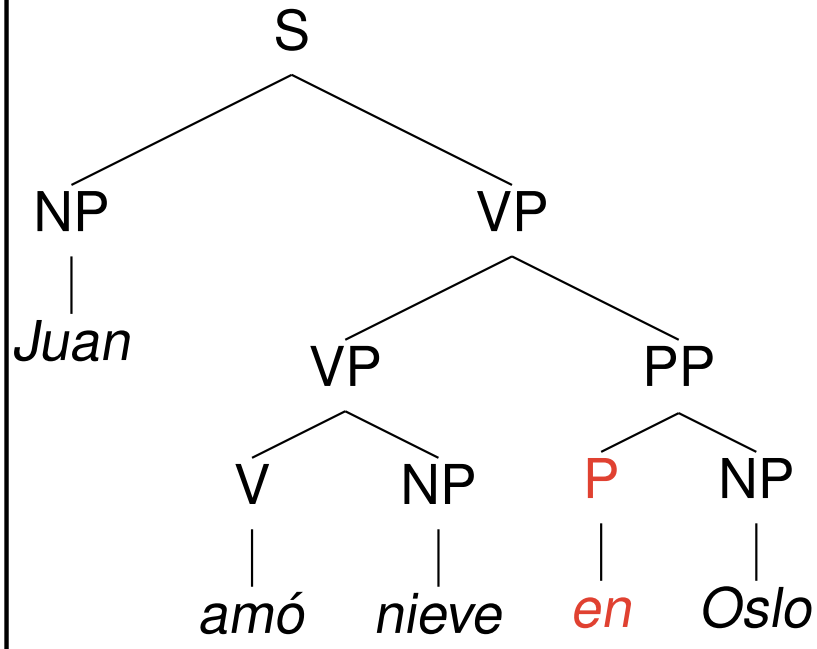
NP → “nieve”

NP → “Juan”

NP → “Oslo”

V → “amó”

P → “en”




*Juan amó nieve en Oslo*

# A Grossly Simplified Example


## The Grammar of Spanish

S → NP VP

VP → V NP

VP → VP PP

PP → P NP

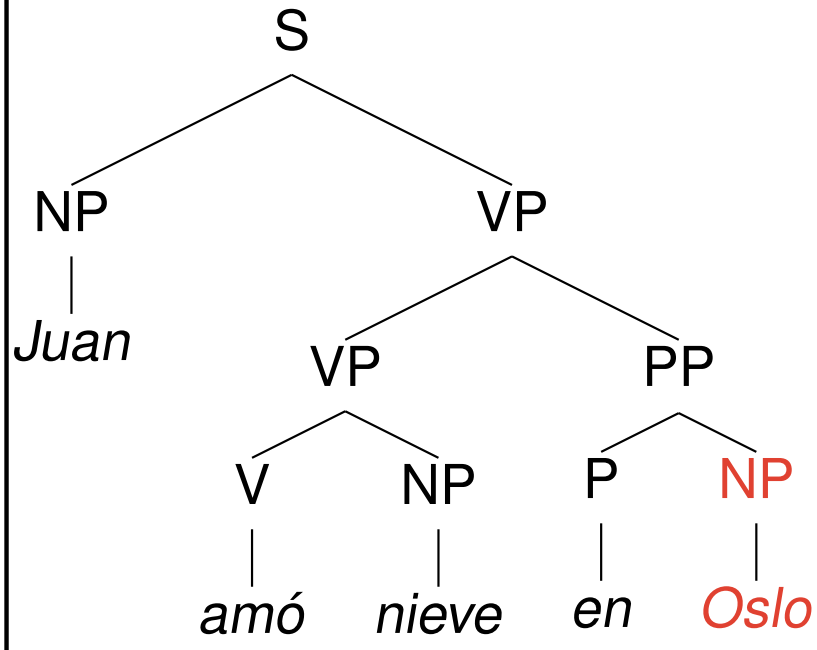
NP → “nieve”

NP → “Juan”

NP → “Oslo”

V → “amó”

P → “en”

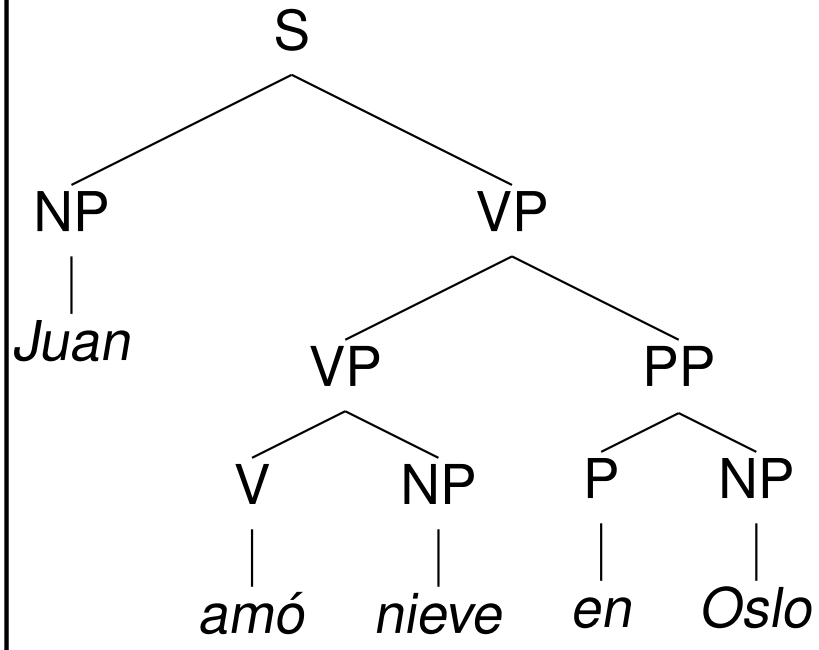



*Juan amó nieve en Oslo*

# A Grossly Simplified Example

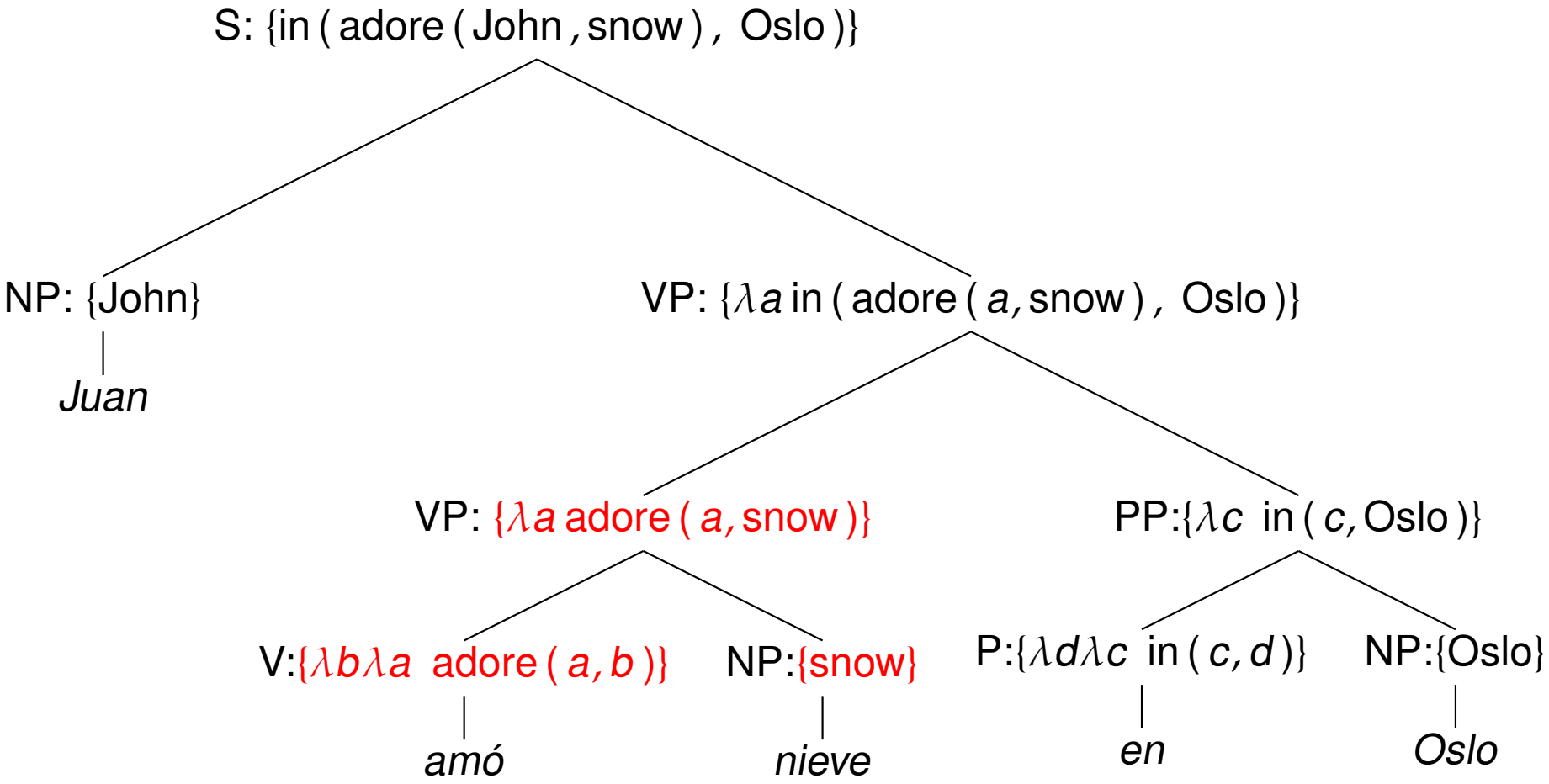

## The Grammar of Spanish

| | |
|---|---|
| $S \rightarrow NP\ VP$ | $\{VP(NP)\}$ |
| $VP \rightarrow V\ NP$ | $\{V(NP)\}$ |
| $VP \rightarrow VP\ PP$ | $\{PP(VP)\}$ |
| $PP \rightarrow P\ NP$ | $\{P(NP)\}$ |
| NP → “nieve” | {snow} |
| NP → “Juan” | {John} |
| NP → “Oslo” | {Oslo} |
| V → “amó” | $\{\lambda b \lambda a\ \text{adore}(a,b)\}$ |
| P → “en” | $\{\lambda d \lambda c\ \text{in}(c,d)\}$ |

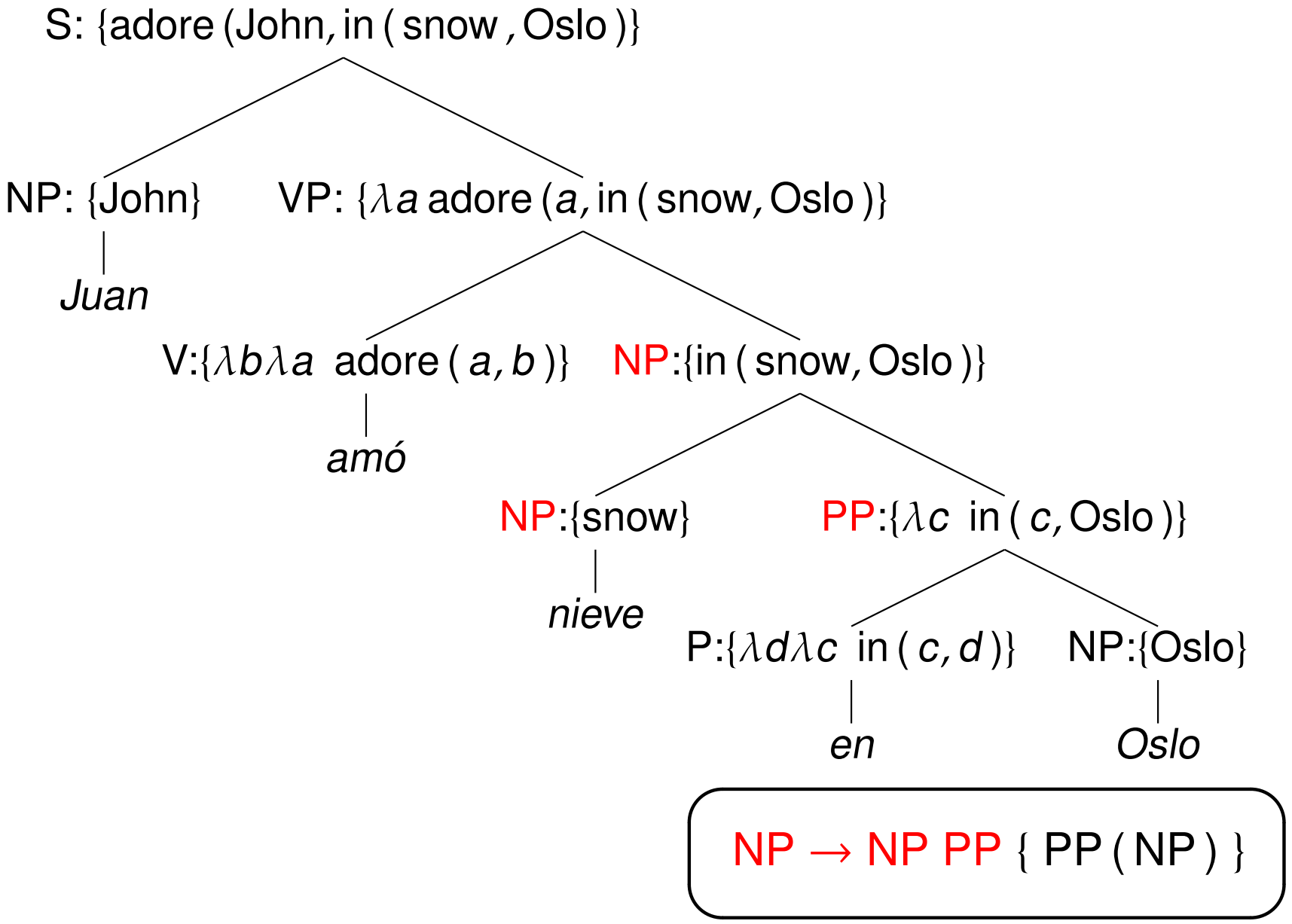



*Juan amó nieve en Oslo*

# Meaning Composition (Still Very Simplified)



# Another Interpretation



# Context Free Grammars (CFGs)


- Formal system for modeling constituent structure.
- Defined in terms of a lexicon and a set of rules

# Context Free Grammars (CFGs)


- Formal system for modeling constituent structure.
- Defined in terms of a lexicon and a set of rules
- Formal models of ‘language’ in a broad sense
  - natural languages, programming languages, communication protocols, . . .

# Context Free Grammars (CFGs)


- Formal system for modeling constituent structure.
- Defined in terms of a lexicon and a set of rules
- Formal models of ‘language’ in a broad sense
  - natural languages, programming languages, communication protocols, . . .
- Can be expressed in the ‘meta-syntax’ of the Backus-Naur Form (BNF) formalism.
  - When looking up concepts and syntax in the Common Lisp HyperSpec, you have been reading (extended) BNF.

# Context Free Grammars (CFGs)


- Formal system for modeling constituent structure.
- Defined in terms of a lexicon and a set of rules
- Formal models of ‘language’ in a broad sense
  - natural languages, programming languages, communication protocols, . . .
- Can be expressed in the ‘meta-syntax’ of the Backus-Naur Form (BNF) formalism.
  - When looking up concepts and syntax in the Common Lisp HyperSpec, you have been reading (extended) BNF.
- Powerful enough to express sophisticated relations among words, yet in a computationally tractable way.

# CFGs (Formally, this Time)


Formally, a CFG is a quadruple: $G = \langle C, \Sigma, P, S \rangle$

# CFGs (Formally, this Time)


Formally, a CFG is a quadruple: $G = \langle C, \Sigma, P, S \rangle$

- $C$ is the set of categories (aka *non-terminals*),
  - $\{\mathrm{S}, \mathrm{NP}, \mathrm{VP}, \mathrm{V}\}$

# CFGs (Formally, this Time)


Formally, a CFG is a quadruple: $G = \langle C, \Sigma, P, S \rangle$

- $C$ is the set of categories (aka *non-terminals*),
  - $\{\mathrm{S}, \mathrm{NP}, \mathrm{VP}, \mathrm{V}\}$
- $\Sigma$ is the vocabulary (aka *terminals*),
  - $\{\mathrm{Kim}, \mathrm{snow}, \mathrm{adores}, \mathrm{in}\}$

# CFGs (Formally, this Time)


Formally, a CFG is a quadruple: $G = \langle C, \Sigma, P, S \rangle$

- $C$ is the set of categories (aka *non-terminals*),
  - $\{\mathrm{S}, \mathrm{NP}, \mathrm{VP}, \mathrm{V}\}$
- $\Sigma$ is the vocabulary (aka *terminals*),
  - $\{\mathrm{Kim}, \mathrm{snow}, \mathrm{adores}, \mathrm{in}\}$
- $P$ is a set of category rewrite rules (aka *productions*)

$$\mathrm{S} \rightarrow \mathrm{NP\ VP} \qquad \mathrm{NP} \rightarrow \mathrm{Kim}$$
$$\mathrm{VP} \rightarrow \mathrm{V\ NP} \qquad \mathrm{NP} \rightarrow \mathrm{snow}$$
$$\mathrm{V} \rightarrow \mathrm{adores}$$

# CFGs (Formally, this Time)


Formally, a CFG is a quadruple: $G = \langle C, \Sigma, P, S \rangle$

- $C$ is the set of categories (aka *non-terminals*),
  - $\{\mathrm{S}, \mathrm{NP}, \mathrm{VP}, \mathrm{V}\}$
- $\Sigma$ is the vocabulary (aka *terminals*),
  - $\{\mathrm{Kim}, \mathrm{snow}, \mathrm{adores}, \mathrm{in}\}$
- $P$ is a set of category rewrite rules (aka *productions*)

$$\begin{array}{ll} \mathrm{S} \rightarrow \mathrm{NP\ VP} & \mathrm{NP} \rightarrow \mathrm{Kim} \\ \mathrm{VP} \rightarrow \mathrm{V\ NP} & \mathrm{NP} \rightarrow \mathrm{snow} \\ & \mathrm{V} \rightarrow \mathrm{adores} \end{array}$$

- $S \in C$ is the *start symbol*, a filter on complete results;

# CFGs (Formally, this Time)


Formally, a CFG is a quadruple: $G = \langle C, \Sigma, P, S \rangle$

- $C$ is the set of categories (aka *non-terminals*),
  - $\{\mathrm{S}, \mathrm{NP}, \mathrm{VP}, \mathrm{V}\}$
- $\Sigma$ is the vocabulary (aka *terminals*),
  - $\{\mathrm{Kim}, \mathrm{snow}, \mathrm{adores}, \mathrm{in}\}$
- $P$ is a set of category rewrite rules (aka *productions*)

$$\begin{array}{ll} \mathrm{S} \rightarrow \mathrm{NP\ VP} & \mathrm{NP} \rightarrow \mathrm{Kim} \\ \mathrm{VP} \rightarrow \mathrm{V\ NP} & \mathrm{NP} \rightarrow \mathrm{snow} \\ & \mathrm{V} \rightarrow \mathrm{adores} \end{array}$$

- $S \in C$ is the *start symbol*, a filter on complete results;
- for each rule $\alpha \rightarrow \beta_1, \beta_2, ..., \beta_n \in P$: $\alpha \in C$ and $\beta_i \in C \cup \Sigma$

# Generative Grammar


Top-down view of generative grammars:

- For a grammar $G$, the language $\mathcal{L}_G$ is defined as the set of strings that can be derived from $S$.
- To derive $w_1^n$ from $S$, we use the rules in $P$ to recursively rewrite $S$ into the sequence $w_1^n$ where each $w_i \in \Sigma$

# Generative Grammar


Top-down view of generative grammars:

- For a grammar $G$, the language $\mathcal{L}_G$ is defined as the set of strings that can be derived from $S$.
- To derive $w_1^n$ from $S$, we use the rules in $P$ to recursively rewrite $S$ into the sequence $w_1^n$ where each $w_i \in \Sigma$
- The grammar is seen as generating strings.
- *Grammatical* strings are defined as strings that can be generated by the grammar.

# Generative Grammar


Top-down view of generative grammars:

- For a grammar $G$, the language $\mathcal{L}_G$ is defined as the set of strings that can be derived from $S$.
- To derive $w_1^n$ from $S$, we use the rules in $P$ to recursively rewrite $S$ into the sequence $w_1^n$ where each $w_i \in \Sigma$
- The grammar is seen as generating strings.
- *Grammatical* strings are defined as strings that can be generated by the grammar.
- The 'context-freeness' of CFGs refers to the fact that we rewrite non-terminals without regard to the overall context in which they occur.

# Treebanks


**Generally**

- A *treebank* is a corpus paired with ‘gold-standard’ (syntactico-semantic) analyses
- Can be created by manual annotation or selection among outputs from automated processing (plus correction).

# Treebanks

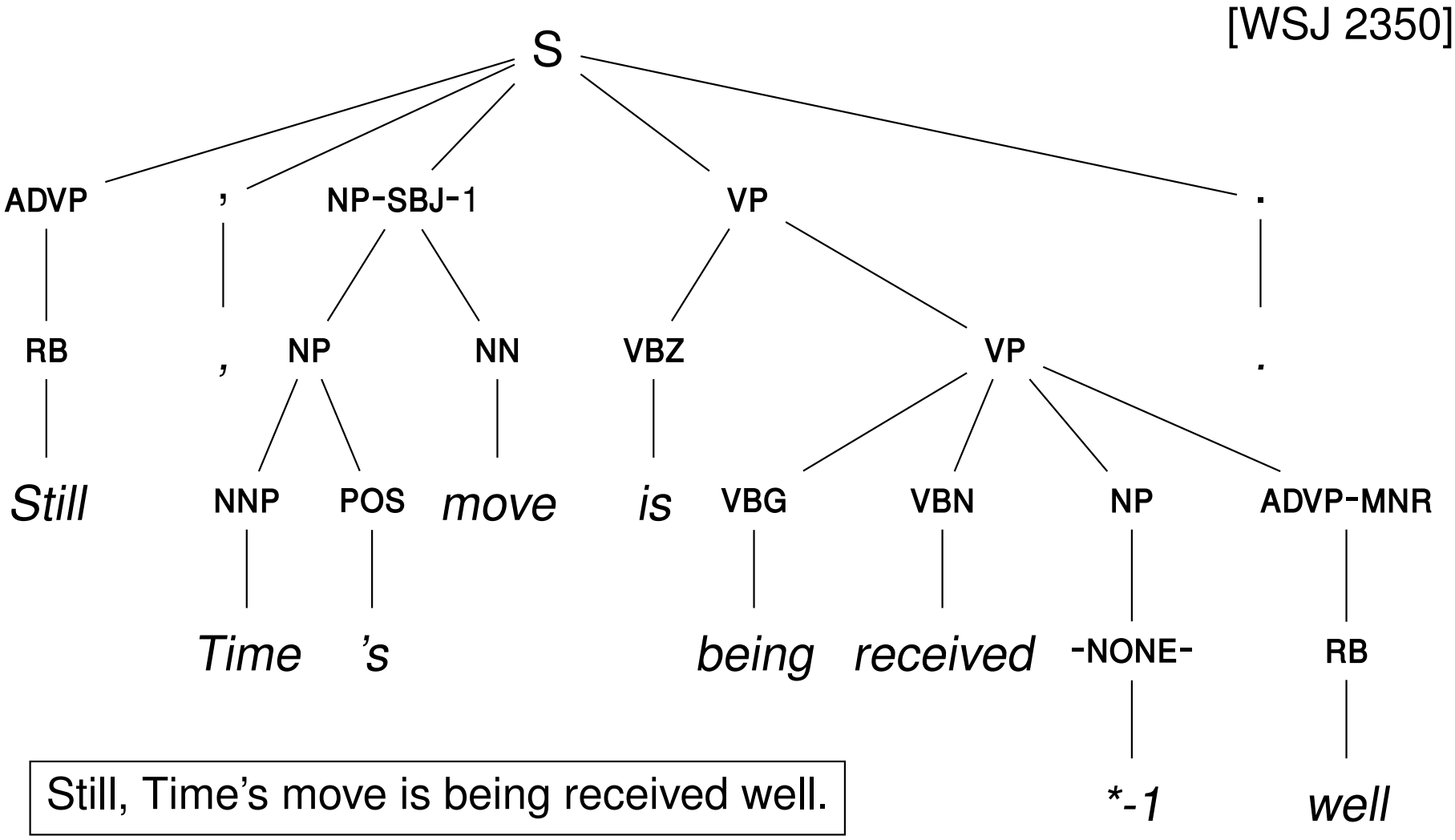

**Generally**

- A *treebank* is a corpus paired with 'gold-standard' (syntactico-semantic) analyses
- Can be created by manual annotation or selection among outputs from automated processing (plus correction).

**Penn Treebank (Marcus et al., 1993)**

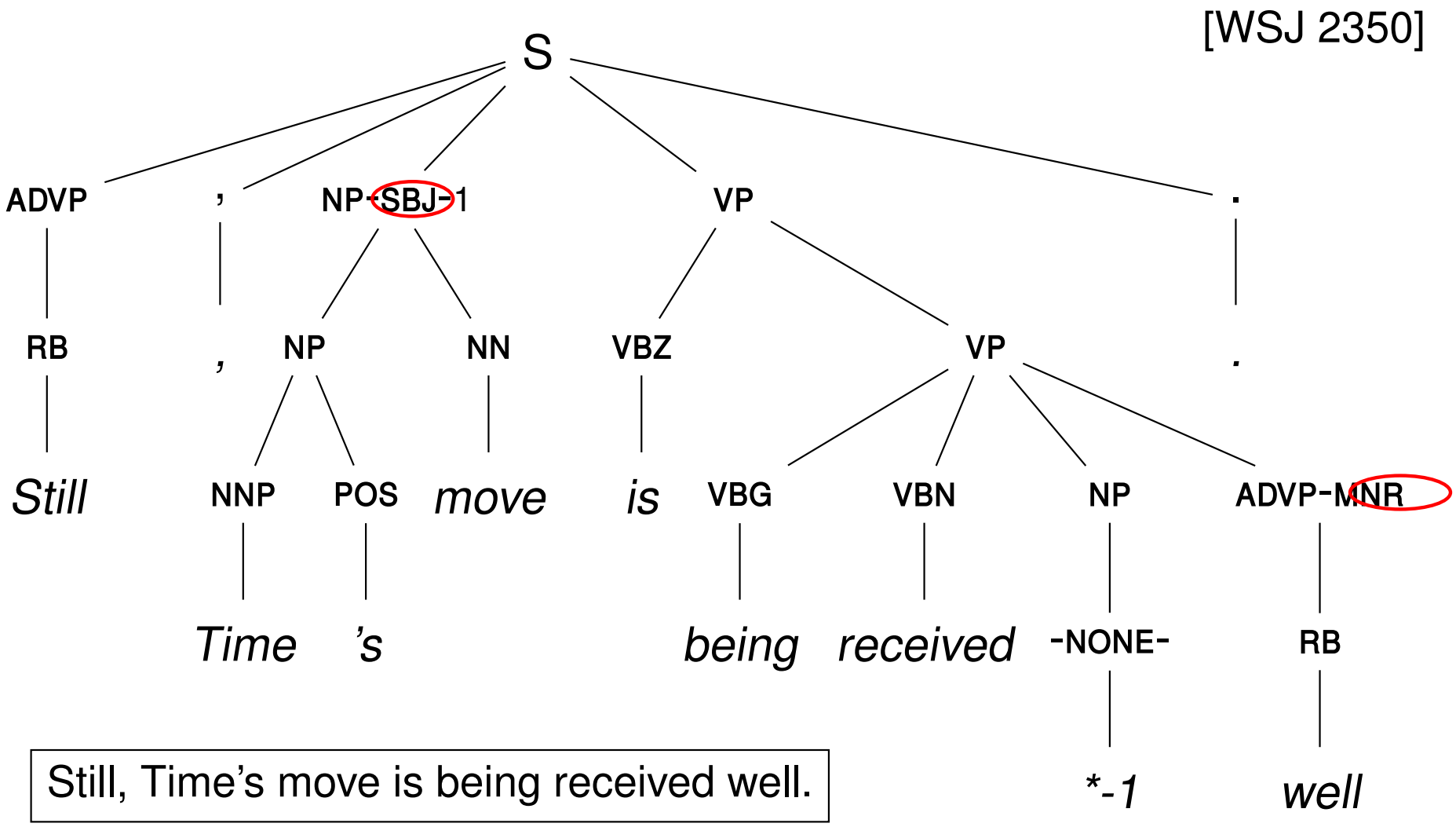
- About one million tokens of Wall Street Journal text
- Hand-corrected PoS annotation using 45 word classes
- Manual annotation with (somewhat) coarse constituent structure

# One Example from the Penn Treebank



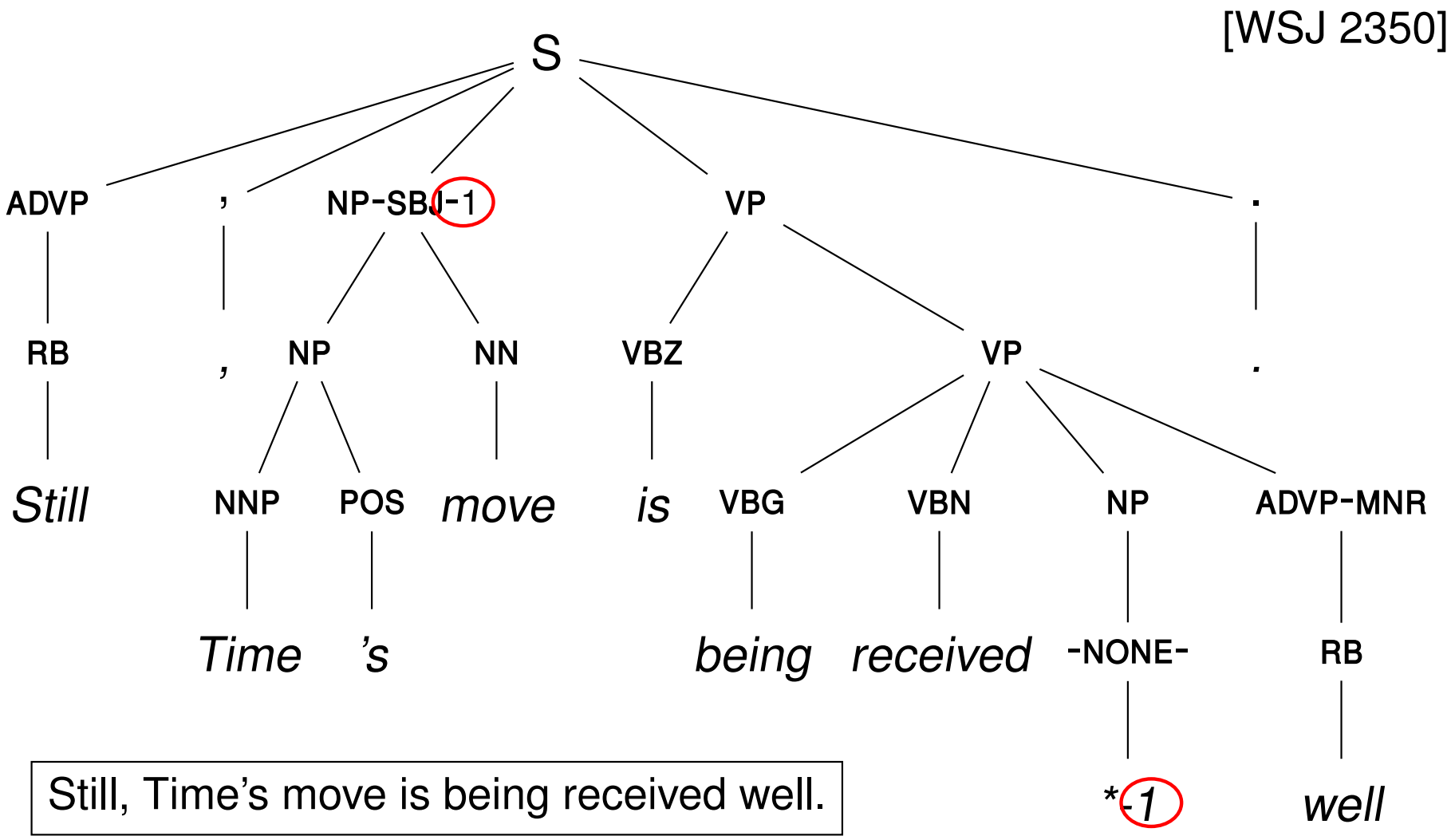

Still, Time’s move is being received well.

# One Example from the Penn Treebank




Still, Time’s move is being received well.

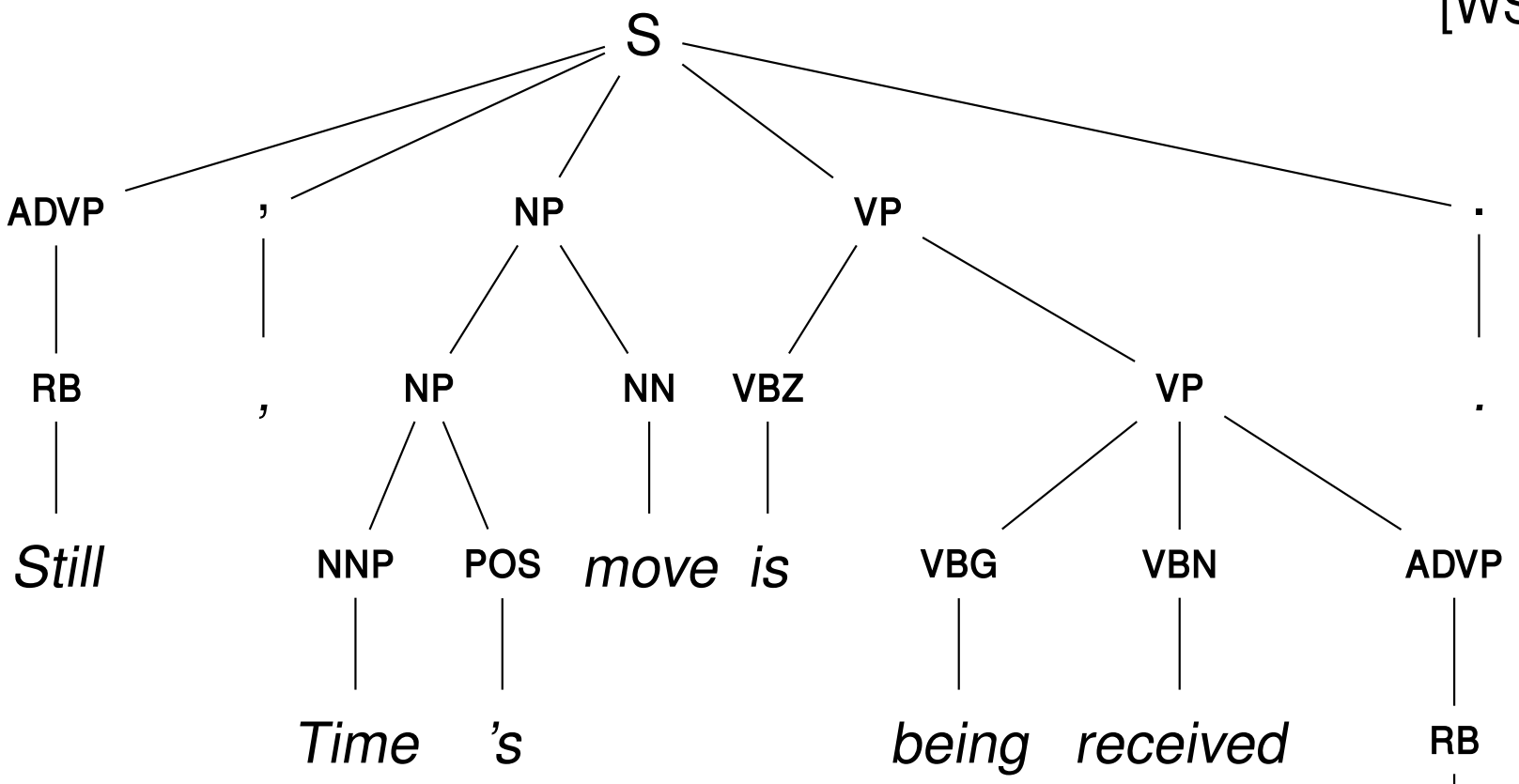
# One Example from the Penn Treebank




Still, Time's move is being received well.

# Elimination of Traces and Functions

[WSJ 2350]




Still, Time’s move is being received well.

# Probabilitic Context-Free Grammars


- We are interested, not just in which trees apply to a sentence, but also to which tree is most likely.

# Probabilitic Context-Free Grammars

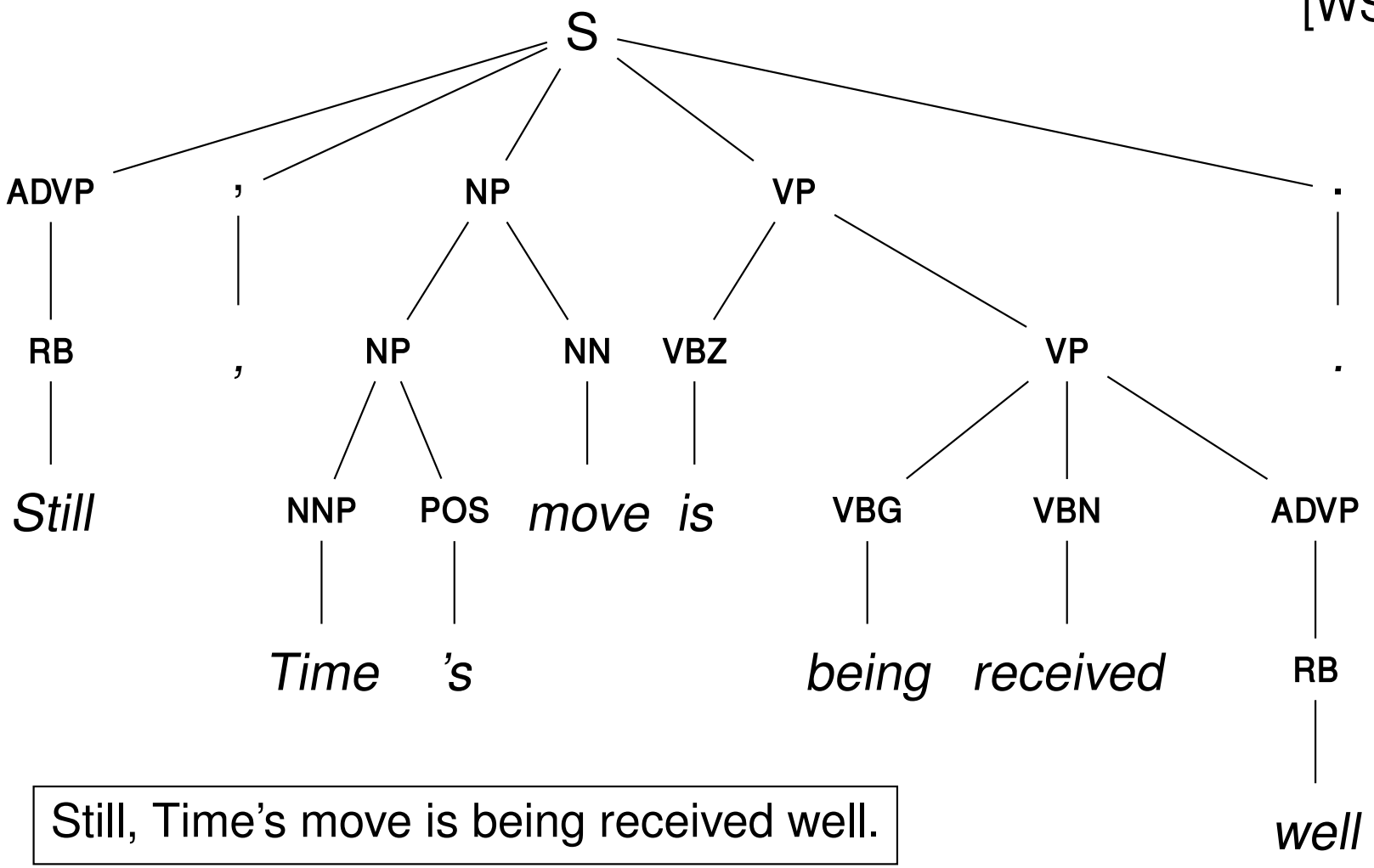

- We are interested, not just in which trees apply to a sentence, but also to which tree is most likely.
- Probabilistic context-free grammars (PCFGs) augment CFGs by adding probabilities to each production, e.g.
  - $S \rightarrow NP\ VP$ 0.6
  - $S \rightarrow NP\ VP\ PP$ 0.4
- These are conditional probabilities — the probability of the right hand side (RHS) given the left hand side (LHS)
  - $P(S \rightarrow NP\ VP) = P(NP\ VP|S)$

# Probabilitic Context-Free Grammars


- We are interested, not just in which trees apply to a sentence, but also to which tree is most likely.
- Probabilistic context-free grammars (PCFGs) augment CFGs by adding probabilities to each production, e.g.
  - S → NP VP 0.6
  - S → NP VP PP 0.4
- These are conditional probabilities — the probability of the right hand side (RHS) given the left hand side (LHS)
  - P(S → NP VP) = P(NP VP|S)
- We can learn these probabilities from a treebank, again using Maximum Likelihood Estimation.

# Estimating PCFGs (1/3)

[WSJ 2350]




Still, Time’s move is being received well.

# Estimating PCFGs (2/3)


```
(S
   (ADVP (RB "Still"))
   (|,| ",")
   (NP
      (NP (NNP "Time") (POS "'s"))
      (NN "move"))
    (VP
      (VBZ "is")
      (VP
         (VBG "being")
         (VP
           (VBN "received")
           (ADVP (RB "well")))))
   (\. "."))
```

# Estimating PCFGs (2/3)

RB → Still 1

```
(S
   (ADVP (RB "Still"))
   (|,| ",")
   (NP
      (NP (NNP "Time") (POS "'s"))
      (NN "move"))
    (VP
      (VBZ "is")
      (VP
         (VBG "being")
         (VP
           (VBN "received")
           (ADVP (RB "well")))))
   (\. "."))
```

# Estimating PCFGs (2/3)

| | |
|---|---|
| RB → Still | 1 |
| ADVP → RB | 1 |

```
(S
   (ADVP (RB "Still"))
   (|,| ",")
   (NP
      (NP (NNP "Time") (POS "'s"))
      (NN "move"))
    (VP
      (VBZ "is")
      (VP
         (VBG "being")
         (VP
           (VBN "received")
           (ADVP (RB "well")))))
   (\. "."))
```

# Estimating PCFGs (2/3)


```
(S
   (ADVP (RB "Still"))
   (|,| ",")
   (NP
      (NP (NNP "Time") (POS "'s"))
      (NN "move"))
    (VP
      (VBZ "is")
      (VP
         (VBG "being")
         (VP
           (VBN "received")
           (ADVP (RB "well")))))
   (\. "."))
```

| | |
|---|---|
| RB → Still | 1 |
| ADVP → RB | 1 |
| \|,\| → , | 1 |

# Estimating PCFGs (2/3)


```
(S
   (ADVP (RB "Still"))
   (|,| ",")
   (NP
      (NP (NNP "Time") (POS "'s"))
      (NN "move"))
    (VP
      (VBZ "is")
      (VP
         (VBG "being")
         (VP
           (VBN "received")
           (ADVP (RB "well")))))
   (\. "."))
```

| | |
|---|---|
| RB → Still | 1 |
| ADVP → RB | 1 |
| \|,\| → , | 1 |
| NNP → Time | 1 |

# Estimating PCFGs (2/3)


```
(S
   (ADVP (RB "Still"))
   (|,| ",")
   (NP
      (NP (NNP "Time") (POS "'s"))
      (NN "move"))
    (VP
      (VBZ "is")
      (VP
         (VBG "being")
         (VP
           (VBN "received")
           (ADVP (RB "well")))))
   (\. "."))
```

| | |
|---|---|
| RB → Still | 1 |
| ADVP → RB | 1 |
| \|,\| → , | 1 |
| NNP → Time | 1 |
| POS → 's | 1 |

# Estimating PCFGs (2/3)


```
(S
   (ADVP (RB "Still"))
   (|,| ",")
   (NP
      (NP (NNP "Time") (POS "'s"))
      (NN "move"))
    (VP
      (VBZ "is")
      (VP
         (VBG "being")
         (VP
           (VBN "received")
           (ADVP (RB "well")))))
   (\. "."))
```

| | |
|---|---|
| RB → Still | 1 |
| ADVP → RB | 1 |
| \|,\| → , | 1 |
| NNP → Time | 1 |
| POS → 's | 1 |
| NP → NNP POS | 1 |

# Estimating PCFGs (2/3)


```
(S
   (ADVP (RB "Still"))
   (|,| ",")
   (NP
      (NP (NNP "Time") (POS "'s"))
      (NN "move"))
    (VP
      (VBZ "is")
      (VP
         (VBG "being")
         (VP
           (VBN "received")
           (ADVP (RB "well")))))
   (\. "."))
```

| | |
|---|---|
| RB → Still | 1 |
| ADVP → RB | 1 |
| \|,\| → , | 1 |
| NNP → Time | 1 |
| POS → 's | 1 |
| NP → NNP POS | 1 |
| NN → move | 1 |

# Estimating PCFGs (2/3)


```
(S
   (ADVP (RB "Still"))
   (|,| ",")
   (NP
      (NP (NNP "Time") (POS "'s"))
      (NN "move"))
    (VP
      (VBZ "is")
      (VP
         (VBG "being")
         (VP
           (VBN "received")
           (ADVP (RB "well")))))
   (\. "."))
```

| | |
|---|---|
| RB → Still | 1 |
| ADVP → RB | 1 |
| \|,\| → , | 1 |
| NNP → Time | 1 |
| POS → 's | 1 |
| NP → NNP POS | 1 |
| NN → move | 1 |
| NP → NP NN | 1 |

# Estimating PCFGs (2/3)


```
(S
   (ADVP (RB "Still"))
   (|,| ",")
   (NP
      (NP (NNP "Time") (POS "'s"))
      (NN "move"))
    (VP
      (VBZ "is")
      (VP
         (VBG "being")
         (VP
           (VBN "received")
           (ADVP (RB "well")))))
   (\. "."))
```

| | |
|---|---|
| RB → Still | 1 |
| ADVP → RB | 1 |
| \|,\| → , | 1 |
| NNP → Time | 1 |
| POS → 's | 1 |
| NP → NNP POS | 1 |
| NN → move | 1 |
| NP → NP NN | 1 |
| VBZ → is | 1 |

# Estimating PCFGs (2/3)


```
(S
   (ADVP (RB "Still"))
   (|,| ",")
   (NP
      (NP (NNP "Time") (POS "'s"))
      (NN "move"))
    (VP
      (VBZ "is")
      (VP
         (VBG "being")
         (VP
           (VBN "received")
           (ADVP (RB "well")))))
   (\. "."))
```

| | |
|---|---|
| RB → Still | 1 |
| ADVP → RB | 1 |
| \|,\| → , | 1 |
| NNP → Time | 1 |
| POS → 's | 1 |
| NP → NNP POS | 1 |
| NN → move | 1 |
| NP → NP NN | 1 |
| VBZ → is | 1 |
| VBG → being | 1 |

# Estimating PCFGs (2/3)


```
(S
   (ADVP (RB "Still"))
   (|,| ",")
   (NP
      (NP (NNP "Time") (POS "'s"))
      (NN "move"))
    (VP
      (VBZ "is")
      (VP
         (VBG "being")
         (VP
           (VBN "received")
           (ADVP (RB "well")))))
   (\. "."))
```

| | |
|---|---|
| RB → Still | 1 |
| ADVP → RB | 1 |
| \|,\| → , | 1 |
| NNP → Time | 1 |
| POS → 's | 1 |
| NP → NNP POS | 1 |
| NN → move | 1 |
| NP → NP NN | 1 |
| VBZ → is | 1 |
| VBG → being | 1 |
| VBN → received | 1 |

# Estimating PCFGs (2/3)

```
(S
   (ADVP (RB "Still"))
   (|,| ",")
   (NP
      (NP (NNP "Time") (POS "'s"))
      (NN "move"))
    (VP
      (VBZ "is")
      (VP
         (VBG "being")
         (VP
           (VBN "received")
           (ADVP (RB "well")))))
   (\. "."))
```

| | |
|---|---|
| RB → Still | 1 |
| ADVP → RB | 1 |
| \|,\| → , | 1 |
| NNP → Time | 1 |
| POS → 's | 1 |
| NP → NNP POS | 1 |
| NN → move | 1 |
| NP → NP NN | 1 |
| VBZ → is | 1 |
| VBG → being | 1 |
| VBN → received | 1 |
| RB → well | 1 |

# Estimating PCFGs (2/3)


```
(S
   (ADVP (RB "Still"))
   (|,| ",")
   (NP
      (NP (NNP "Time") (POS "'s"))
      (NN "move"))
    (VP
      (VBZ "is")
      (VP
         (VBG "being")
         (VP
           (VBN "received")
           (ADVP (RB "well")))))
   (\. "."))
```

| | |
|---|---|
| RB → Still | 1 |
| ADVP → RB | 2 |
| \|,\| → , | 1 |
| NNP → Time | 1 |
| POS → 's | 1 |
| NP → NNP POS | 1 |
| NN → move | 1 |
| NP → NP NN | 1 |
| VBZ → is | 1 |
| VBG → being | 1 |
| VBN → received | 1 |
| RB → well | 1 |

# Estimating PCFGs (2/3)


```
(S
   (ADVP (RB "Still"))
   (|,| ",")
   (NP
      (NP (NNP "Time") (POS "'s"))
      (NN "move"))
    (VP
      (VBZ "is")
      (VP
         (VBG "being")
         (VP
           (VBN "received")
           (ADVP (RB "well")))))
   (\. "."))
```

| | |
|---|---|
| RB → Still | 1 |
| ADVP → RB | 2 |
| \|,\| → , | 1 |
| NNP → Time | 1 |
| POS → 's | 1 |
| NP → NNP POS | 1 |
| NN → move | 1 |
| NP → NP NN | 1 |
| VBZ → is | 1 |
| VBG → being | 1 |
| VBN → received | 1 |
| RB → well | 1 |
| VP → VBN ADVP | 1 |

# Estimating PCFGs (2/3)


```
(S
   (ADVP (RB "Still"))
   (|,| ",")
   (NP
      (NP (NNP "Time") (POS "'s"))
      (NN "move"))
    (VP
      (VBZ "is")
      (VP
         (VBG "being")
         (VP
           (VBN "received")
           (ADVP (RB "well")))))
   (\. "."))
```

| | |
|---|---|
| RB → Still | 1 |
| ADVP → RB | 2 |
| \|,\| → , | 1 |
| NNP → Time | 1 |
| POS → 's | 1 |
| NP → NNP POS | 1 |
| NN → move | 1 |
| NP → NP NN | 1 |
| VBZ → is | 1 |
| VBG → being | 1 |
| VBN → received | 1 |
| RB → well | 1 |
| VP → VBN ADVP | 1 |
| VP → VBG VP | 1 |

# Estimating PCFGs (2/3)


```
(S
   (ADVP (RB "Still"))
   (|,| ",")
   (NP
      (NP (NNP "Time") (POS "'s"))
      (NN "move"))
    (VP
      (VBZ "is")
      (VP
         (VBG "being")
         (VP
           (VBN "received")
           (ADVP (RB "well")))))
   (\. "."))
```

| | |
|---|---|
| RB → Still | 1 |
| ADVP → RB | 2 |
| \|,\| → , | 1 |
| NNP → Time | 1 |
| POS → 's | 1 |
| NP → NNP POS | 1 |
| NN → move | 1 |
| NP → NP NN | 1 |
| VBZ → is | 1 |
| VBG → being | 1 |
| VBN → received | 1 |
| RB → well | 1 |
| VP → VBN ADVP | 1 |
| VP → VBG VP | 1 |
| \. → . | 1 |

# Estimating PCFGs (2/3)


```
(S
   (ADVP (RB "Still"))
   (|,| ",")
   (NP
      (NP (NNP "Time") (POS "'s"))
      (NN "move"))
    (VP
      (VBZ "is")
      (VP
         (VBG "being")
         (VP
           (VBN "received")
           (ADVP (RB "well")))))
   (\. "."))
```

| | |
|---|---|
| RB → Still | 1 |
| ADVP → RB | 2 |
| \|,\| → , | 1 |
| NNP → Time | 1 |
| POS → 's | 1 |
| NP → NNP POS | 1 |
| NN → move | 1 |
| NP → NP NN | 1 |
| VBZ → is | 1 |
| VBG → being | 1 |
| VBN → received | 1 |
| RB → well | 1 |
| VP → VBN ADVP | 1 |
| VP → VBG VP | 1 |
| \. → . | 1 |
| S → ADVP \|,\| NP VP \. | 1 |

# Estimating PCFGs (2/3)


```
(S
   (ADVP (RB "Still"))
   (|,| ",")
   (NP
      (NP (NNP "Time") (POS "'s"))
      (NN "move"))
    (VP
      (VBZ "is")
      (VP
         (VBG "being")
         (VP
           (VBN "received")
           (ADVP (RB "well")))))
   (\. "."))
```

| | |
|---|---|
| RB → Still | 1 |
| ADVP → RB | 2 |
| \|,\| → , | 1 |
| NNP → Time | 1 |
| POS → 's | 1 |
| NP → NNP POS | 1 |
| NN → move | 1 |
| NP → NP NN | 1 |
| VBZ → is | 1 |
| VBG → being | 1 |
| VBN → received | 1 |
| RB → well | 1 |
| VP → VBN ADVP | 1 |
| VP → VBG VP | 1 |
| \. → . | 1 |
| S → ADVP \|,\| NP VP \. | 1 |
| START → S | 1 |

# Estimating PCFGs (3/3)


Once we have counts of all the rules, we turn them into probabilities.

# Estimating PCFGs (3/3)


Once we have counts of all the rules, we turn them into probabilities.

| | | | |
|---|---|---|---|
| S → ADVP \|,\| NP VP \. | 50 | S → NP VP \. | 400 |
| S → NP VP PP \. | 350 | S → VP ! | 100 |
| S → NP VP S \. | 200 | S → NP VP | 50 |

# Estimating PCFGs (3/3)


Once we have counts of all the rules, we turn them into probabilities.

| | | | |
|---|---|---|---|
| S → ADVP \|,\| NP VP \. | 50 | S → NP VP \. | 400 |
| S → NP VP PP \. | 350 | S → VP ! | 100 |
| S → NP VP S \. | 200 | S → NP VP | 50 |

$$P(S \rightarrow ADVP\ |,|\ NP\ VP\ \backslash.) \approx \frac{C(S \rightarrow ADVP\ |,|\ NP\ VP\ \backslash.)}{C(S)}$$

# Estimating PCFGs (3/3)


Once we have counts of all the rules, we turn them into probabilities.

| | | | |
|---|---|---|---|
| S → ADVP \|,\| NP VP \. | 50 | S → NP VP \. | 400 |
| S → NP VP PP \. | 350 | S → VP ! | 100 |
| S → NP VP S \. | 200 | S → NP VP | 50 |

$$P(S \rightarrow ADVP\ |,|\ NP\ VP\ \backslash.) \approx \frac{C(S \rightarrow ADVP\ |,|\ NP\ VP\ \backslash.)}{C(S)}$$

$$= \frac{50}{1150}$$

$$= 0.0435$$