Probabilistic Top-Down Parsing and Language Modeling

Brian Roark (2001)

Incremental Processing - Vera Demberg
26. Mai 2011
Carolyn Ladda

Overview

- Motivation
- Probabilistic Grammars
- Top-Down Parsing Algorithm
- Evaluation and Application

Motivation

- psycholinguistic evidence indicates (strictly) incremental processing in humans
- algorithms following that structure are more human-like
- possible benefits for performance as well as psycholinguistic understanding

Motivation

- new approach to language modeling for speech recognition
- speech recognition is a real-time application with incremental input
- it requires on-line processing

PCFGs

- a CFG $G = (V, T, P, S^\circ)$ consists of:
  - V: set of non-terminal symbols
  - T: set of terminal symbols
  - P: set of production rules
  - S°: start symbol $\in V$

PCFGs

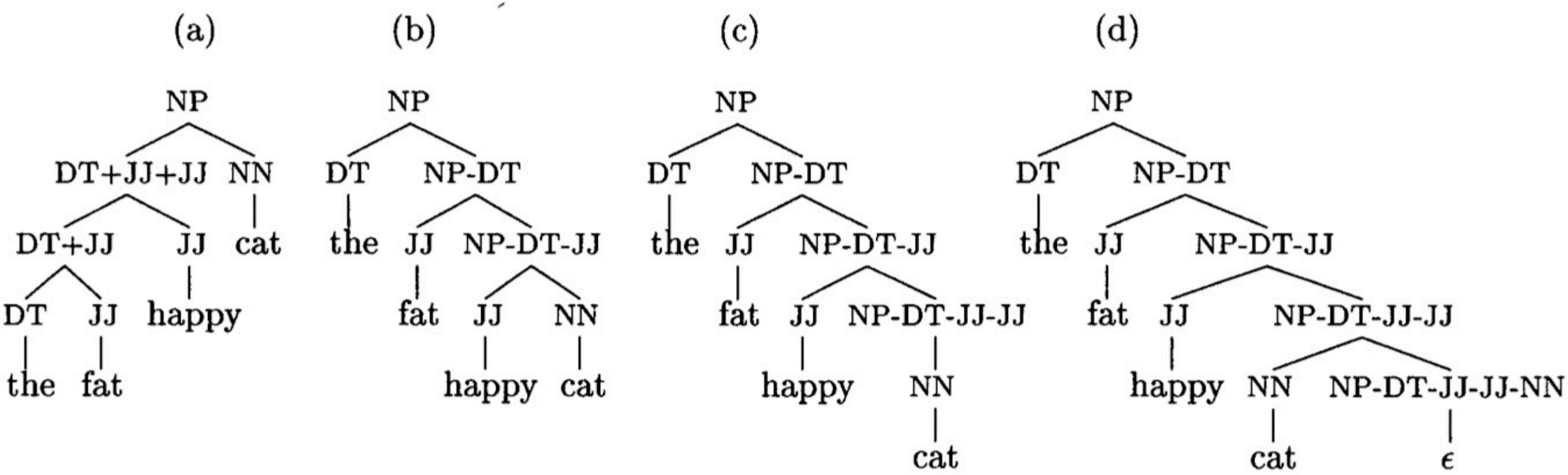
- a CFG *G* defines a language L_G
- a PCFG is a CFG with a probability assigned to each rule
- a probability is assigned to each string

Grammar Transforms

- grammars can be transformed to an equivalent structure in several ways
- this affects the parsing process regarding tree traversal and thus the order of production
- they also depend on the parsing strategy

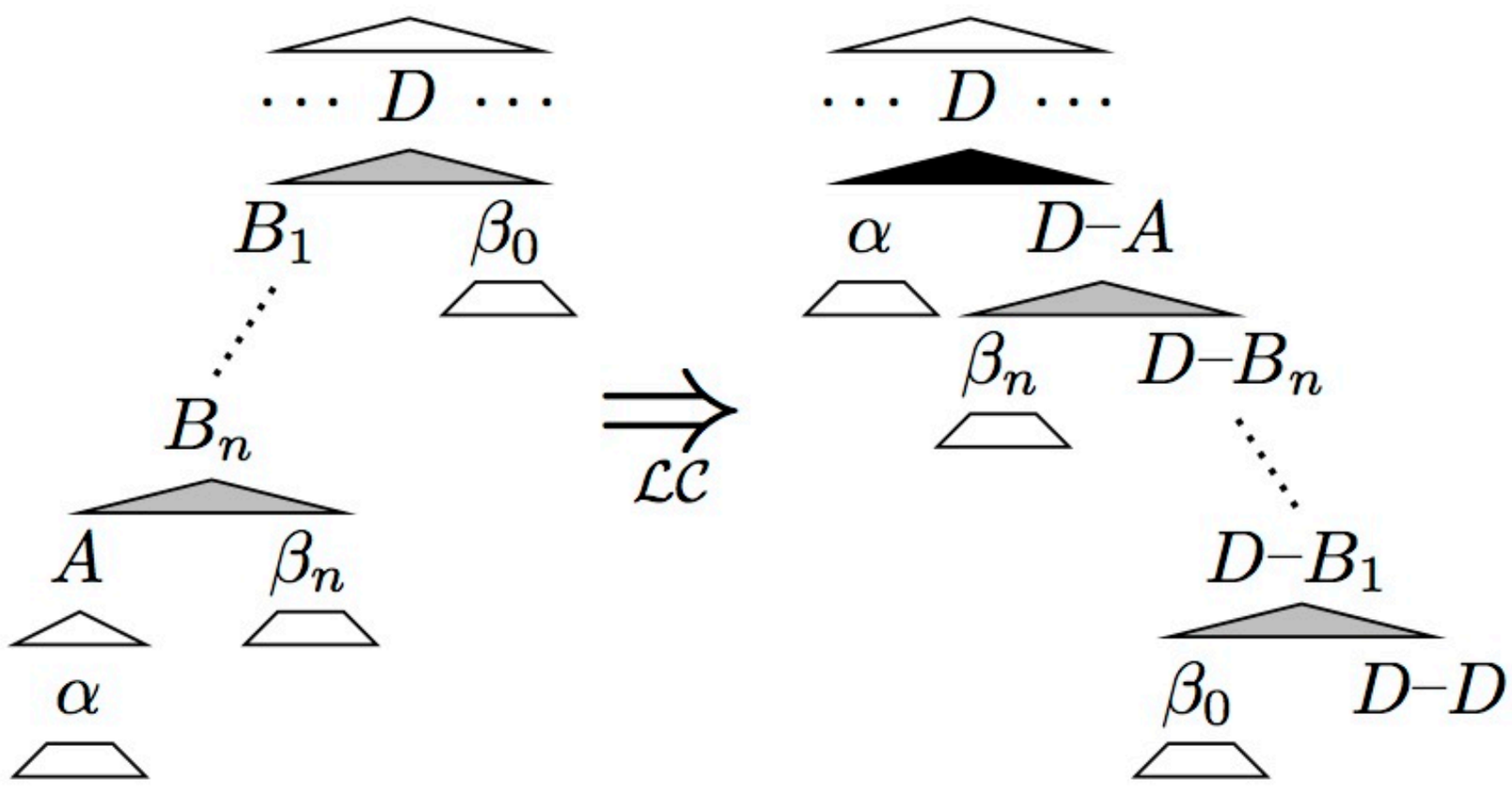
Grammar Transforms

binarized trees



Grammar Transforms

left-corner transform
for left-recursive rules



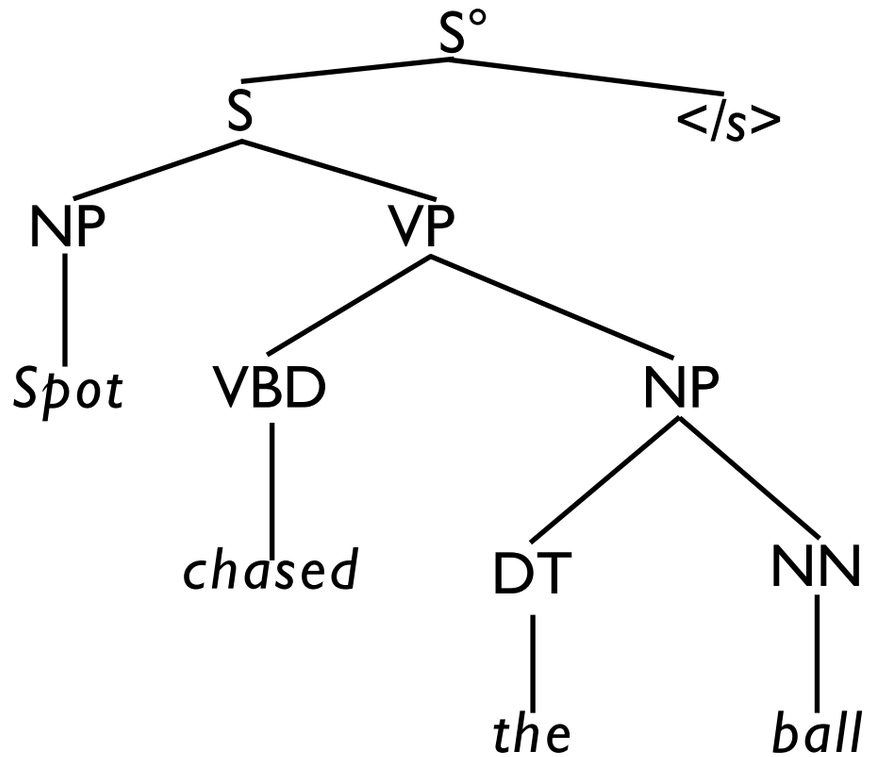
Left-Factorization

- delays predictions about constituents that are expected later in the string
- transforms production rules s.t. all productions are either binary, lexical rules or epsilon productions

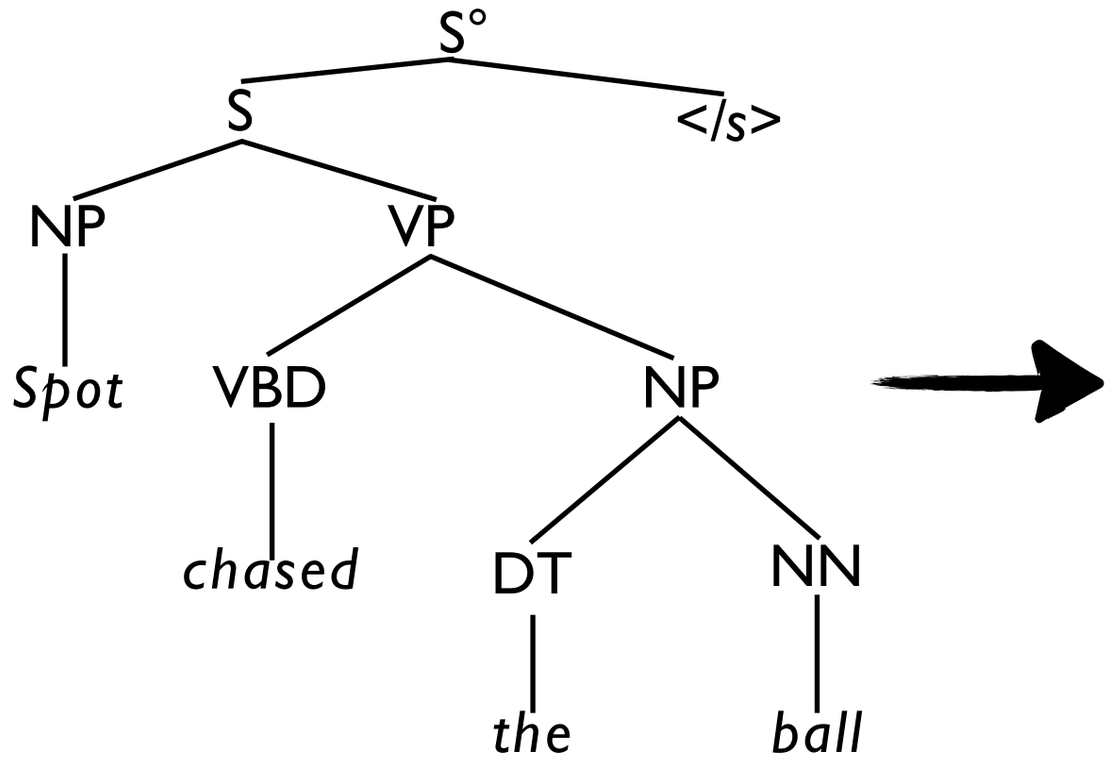
Left-Factorization

- $(A \rightarrow B\ \ A\text{-}B) \in G_f$ iff $(A \rightarrow B\beta) \in G$, s.t. $B \in V$ and $\beta \in V^*$
- $(A\text{-}\alpha \rightarrow B\ \ A\text{-}\alpha B) \in G_f$ iff $(A \rightarrow \alpha B\beta) \in G$, s.t. $B \in V$, $\alpha \in V+$, and $\beta \in V^*$
- $(A\text{-}\alpha B \rightarrow \varepsilon) \in G_f$ iff $(A \rightarrow \alpha B) \in G$, s.t. $B \in V$ and $\alpha \in V^*$
- $(A \rightarrow a) \in G_f$ iff $(A \rightarrow a) \in G$, s.t. $a \in T$

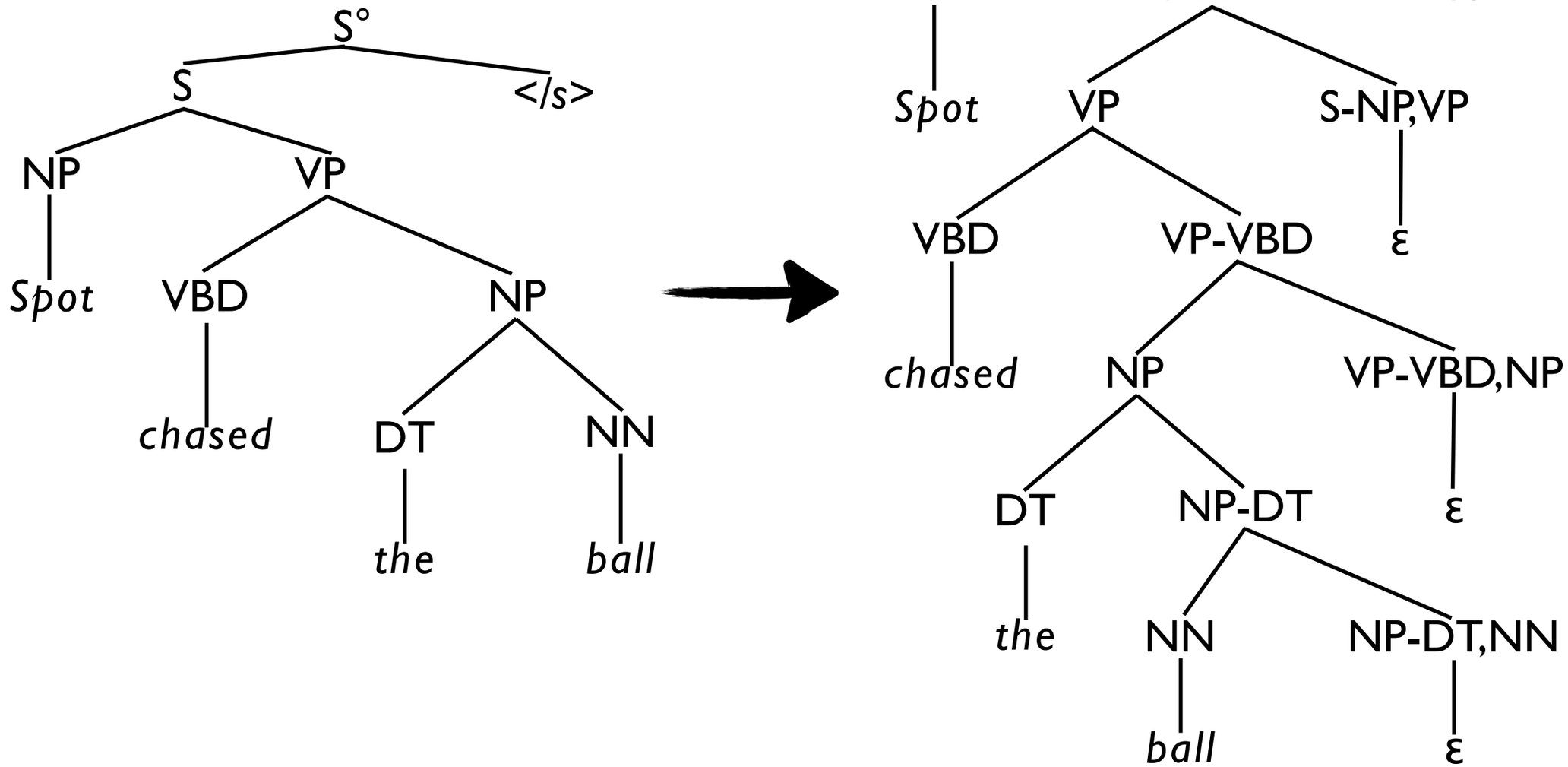
Left-Factorization



Left-Factorization



Left-Factorization

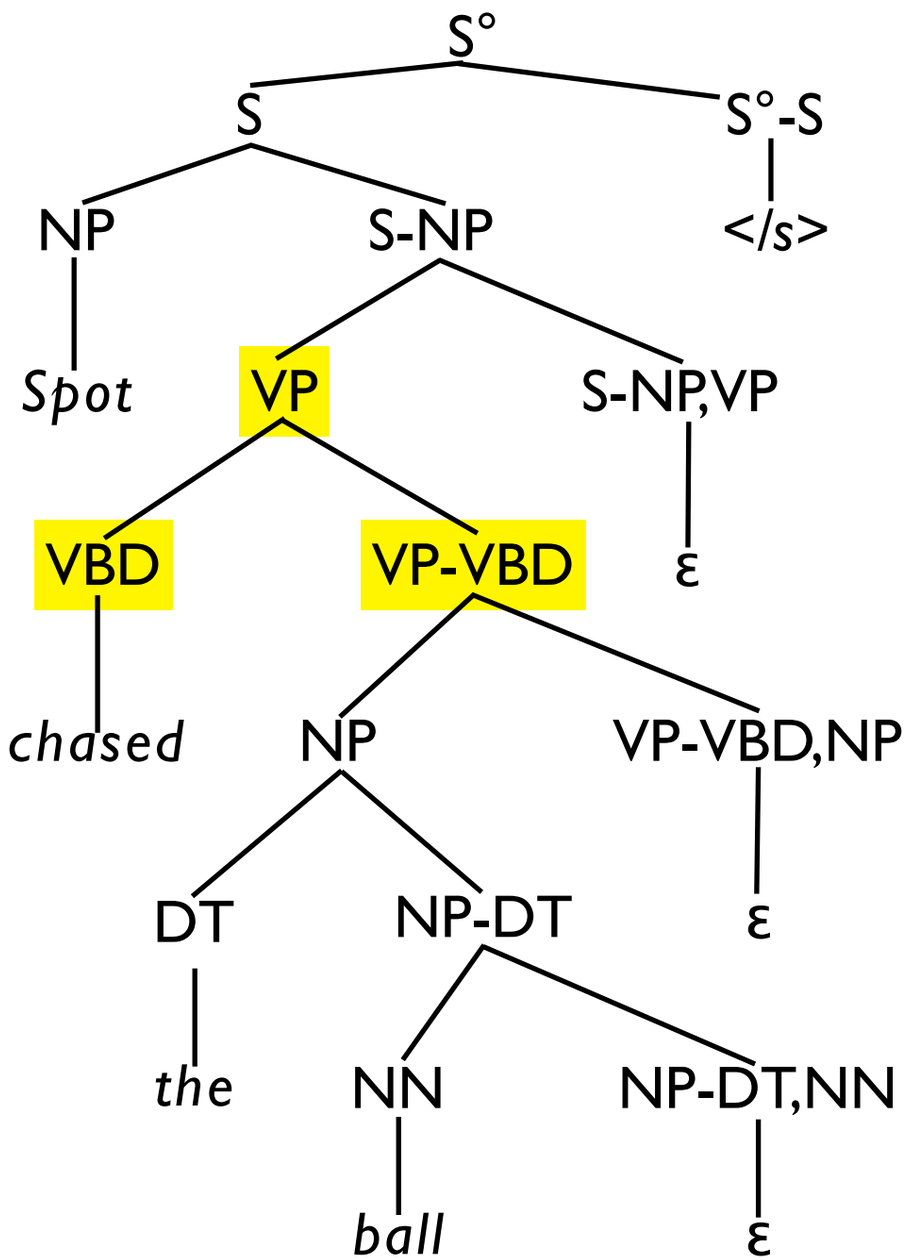


Left-Factorization

$(A \rightarrow B\ \ A\text{-}B) \in G_f$
iff $(A \rightarrow B\beta) \in G$,
s.t. $B \in V$ and $\beta \in V^*$

$VP \rightarrow VBD\ NP \in G$

$VP \rightarrow VBD\ \ VP\text{-}VBD \in G_f$



Left-Factorization

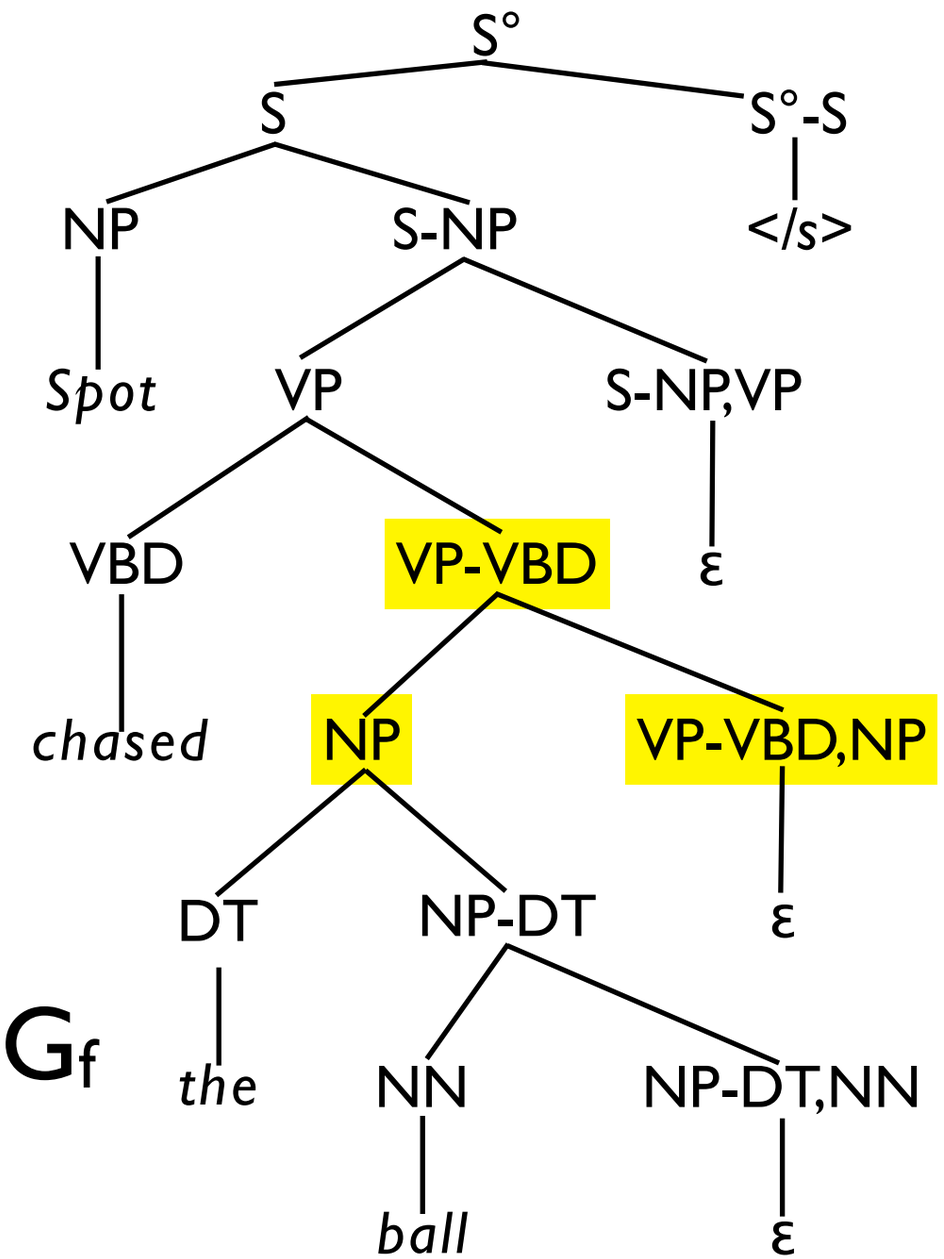
$(A\text{-}\alpha \rightarrow B \;\; A\text{-}\alpha B) \in G_f$

iff $(A \rightarrow \alpha B \beta) \in G$,

s.t. $B \in V$, $\alpha \in V+$, and $\beta \in V^*$

$VP \rightarrow VBD\ NP \in G$

$VP\text{-}VBD \rightarrow NP\ \ VP\text{-}VBD,NP \in G_f$

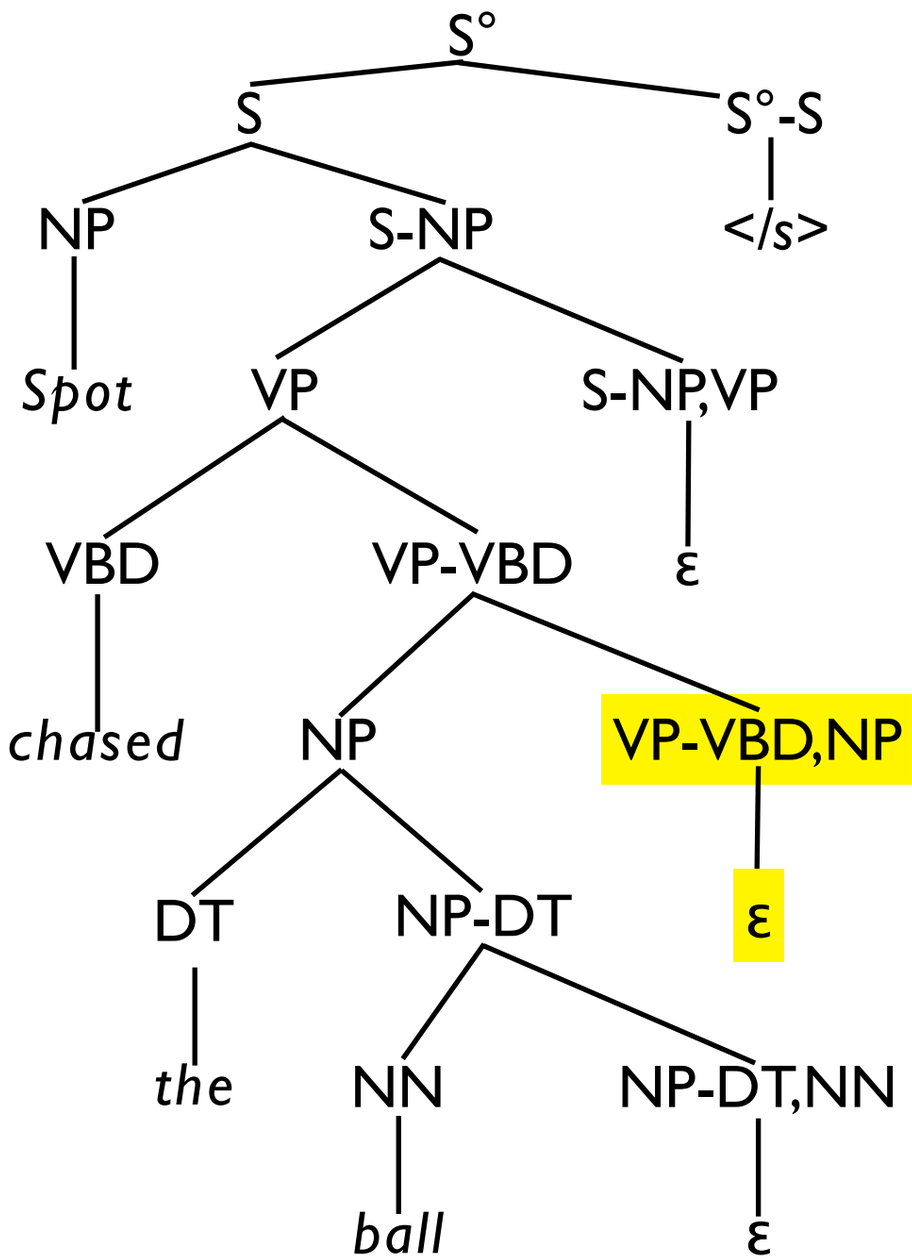


Left-Factorization

$(A\text{-}\alpha B \rightarrow \varepsilon) \in G_f$
iff $(A \rightarrow \alpha B) \in G$,
s.t. $B \in V$ and $\alpha \in V^*$

$VP \rightarrow VBD\ NP \in G$

$VP\text{-}VBD,NP \rightarrow \varepsilon \in G_f$

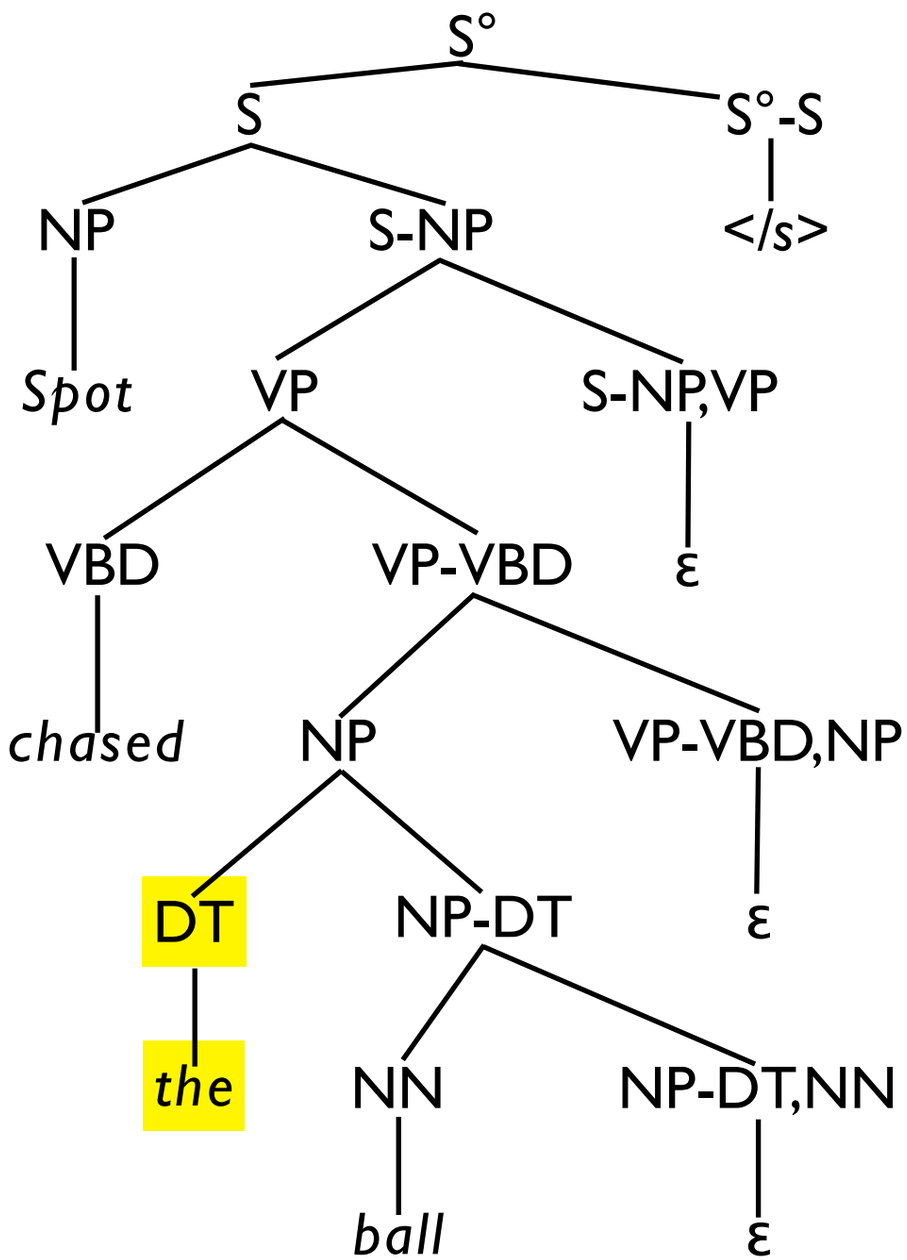


Left-Factorization

$(A \rightarrow a) \in G_f$ iff $(A \rightarrow a) \in G$, s.t. $a \in T$

$DT \rightarrow the \in G$

$DT \rightarrow the \in G_f$



Parsing Algorithm

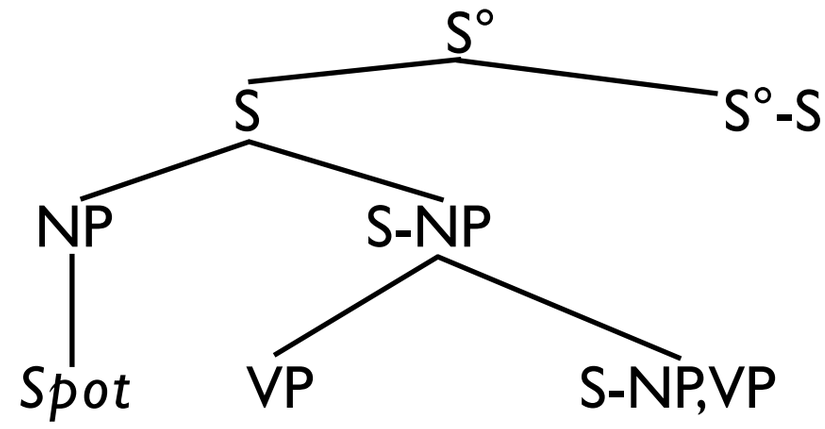
- incremental top-down parser
- beam search, using probability-based heuristics

Parsing Algorithm

- candidate analyses $C = (D, S, P_D, F, w_i^n)$:
  - D: derivation, list of rules
  - S: stack of non-terminal symbols and $
  - P_D: probability
  - F: figure of merit
  - w_i^n : string remaining to be parsed

Parsing Algorithm

- $C = (D, S, P_D, F, w_1)$:
  - D: derivation
  - S: [VP, S-NP,VP, S°-S, $]
  - P_D: p1
  - F: p1 x LAP(S, „chased“)
  - w_1: „chased ...“



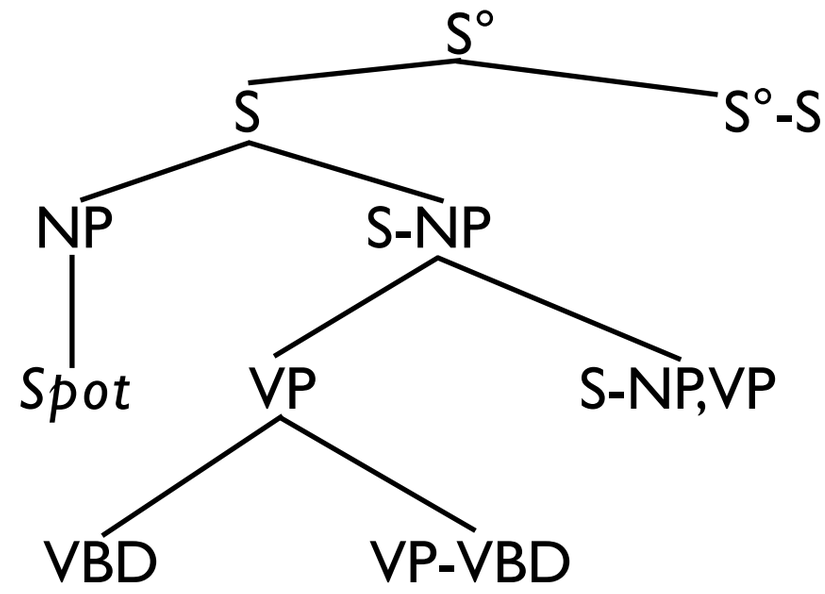
Parsing Algorithm

$$C = (D, S, P_D, F, w_i^n) \Rightarrow C' = (D', S', P_{D'}, F', w_j^n)$$

- $D' = D + A \rightarrow X_o \ldots X_k$
- $S = A\alpha\$$
- *either* $S' = X_o \ldots X_k\alpha\$$ *and* $j = i$
  or $k = 0, X_0 = w_i, j = i + 1$, *and* $S' = \alpha\$$
- $P_{D'} = P_D\, P(A \rightarrow X_o \ldots X_k)$
- $F' = P_{D'}\, LAP(S', w_j)$

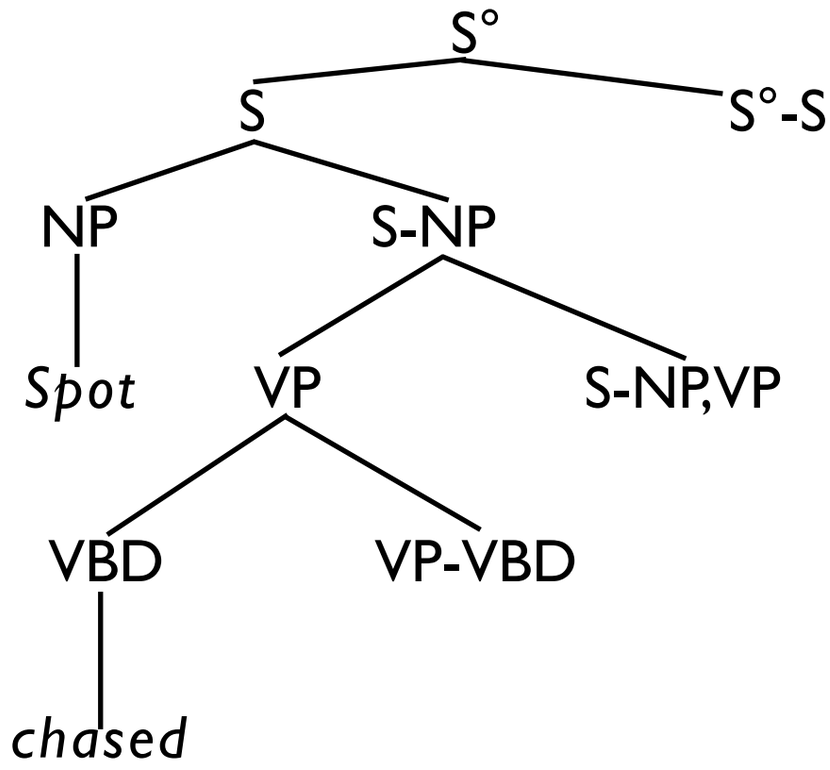
Parsing Algorithm

- $C = (D, S, P_D, F, w_1)$:
  - *D*: derivation
  - *S*: [VBD,VP-VBD, S-NP,VP, S°-S,$]
- P_D: $p_2 = p_1 \times P(VP \rightarrow ..)$
- *F*: $p_2 \times$ LAP(*S*, „chased“)
- w_1: „chased ...“



Parsing Algorithm

- $C = (D, S, P_D, F, w_2)$:
  - D: derivation
  - S: [VP-VBD, S-NP,VP, S°-S,$]
  - P_D: $p_3 = p_2 \times P(\text{VBD} \rightarrow \text{chased})$
  - F: $p_3 \times \text{LAP}(S, \text{„the“})$
  - w_2: „the ...“



Parsing Algorithm

- initial candidate on H_0: *([], S°$, I, I, w_0^n)*
- a candidate is popped from the priority queue H_i
- derived candidates are moved to priority queues H_i and H_{i+1}

Parsing Algorithm

- termination condition: S = *$*, w_i = *</s>*
- if H_{i+1} is sufficiently filled, the algorithm moves from H_i to H_{i+1}
- the complete parse on H_{n+1} with the highest probability is selected as return value

Left-Recursion

- left-factorizing the grammar does not eliminate left-recursion
- due to the nature of beam search improbable candidate analysis are discarded
- the longer a chain of consecutive left-recursive rule productions the more improbable it becomes

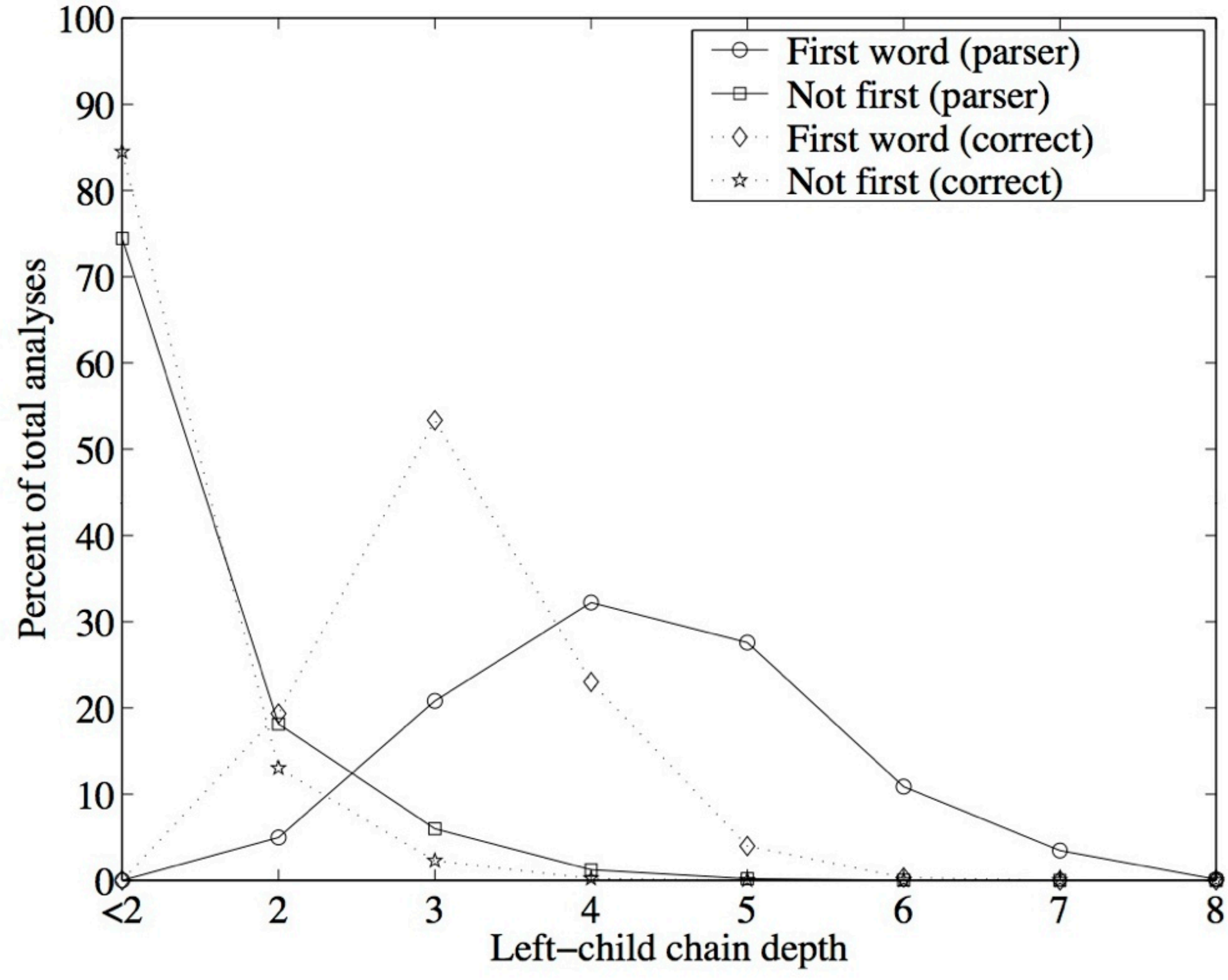
Left-Recursion

- the parser can still be inefficient regarding left-recursion
- left-child chain for a word: consecutive non-terminals above it are the leftmost children within their constituent

Left-Recursion

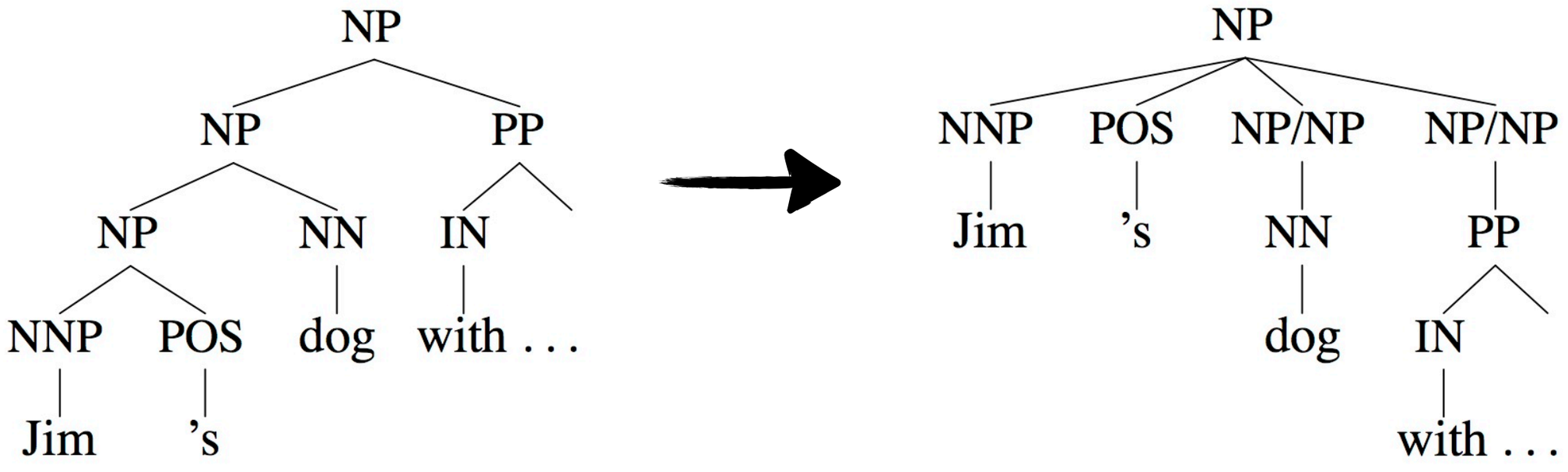
- almost all correct analyses with left-recursive chains are constructed
- many constructed analyses contains chains that are longer than necessary

Left-Recursion



Left-Recursion

- to minimise the amount of incorrect analyses left-recursive NP rules in the grammar are transformed
- a flattened selective left-corner transform is used



Left-Recursion

- recall and precision, as well as parsing time increase slightly compared to an entirely left-factorized grammar
- with an adjusted beam factor, recall and precision are the same and the parser is 40% more efficient

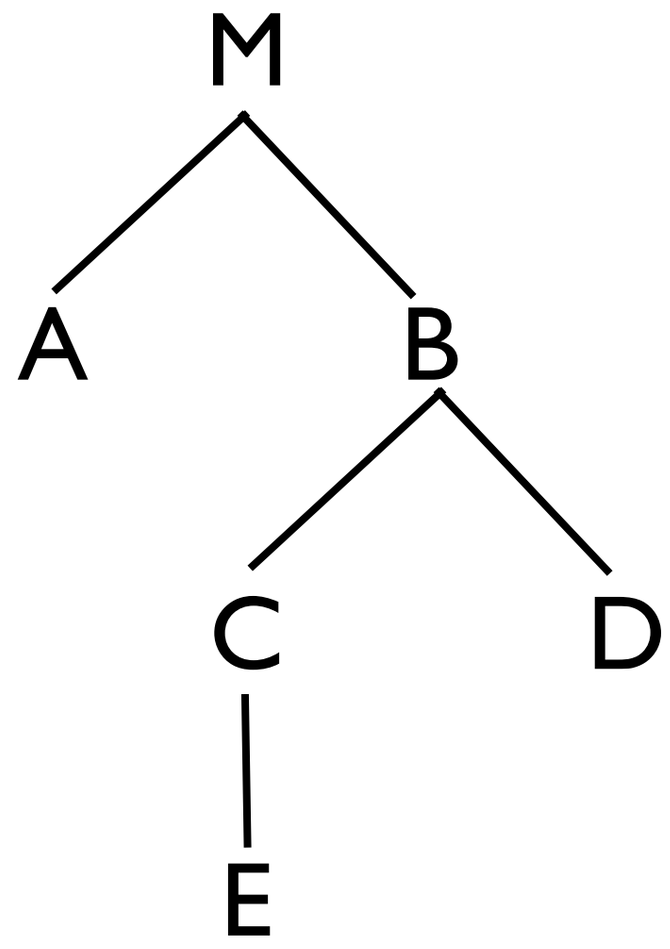
Conditioning

- additionally to the form of an expanding CFG rule other features of a candidate analysis are taken into account
- the incremental parser uses information to predict the nature of further input more accurately

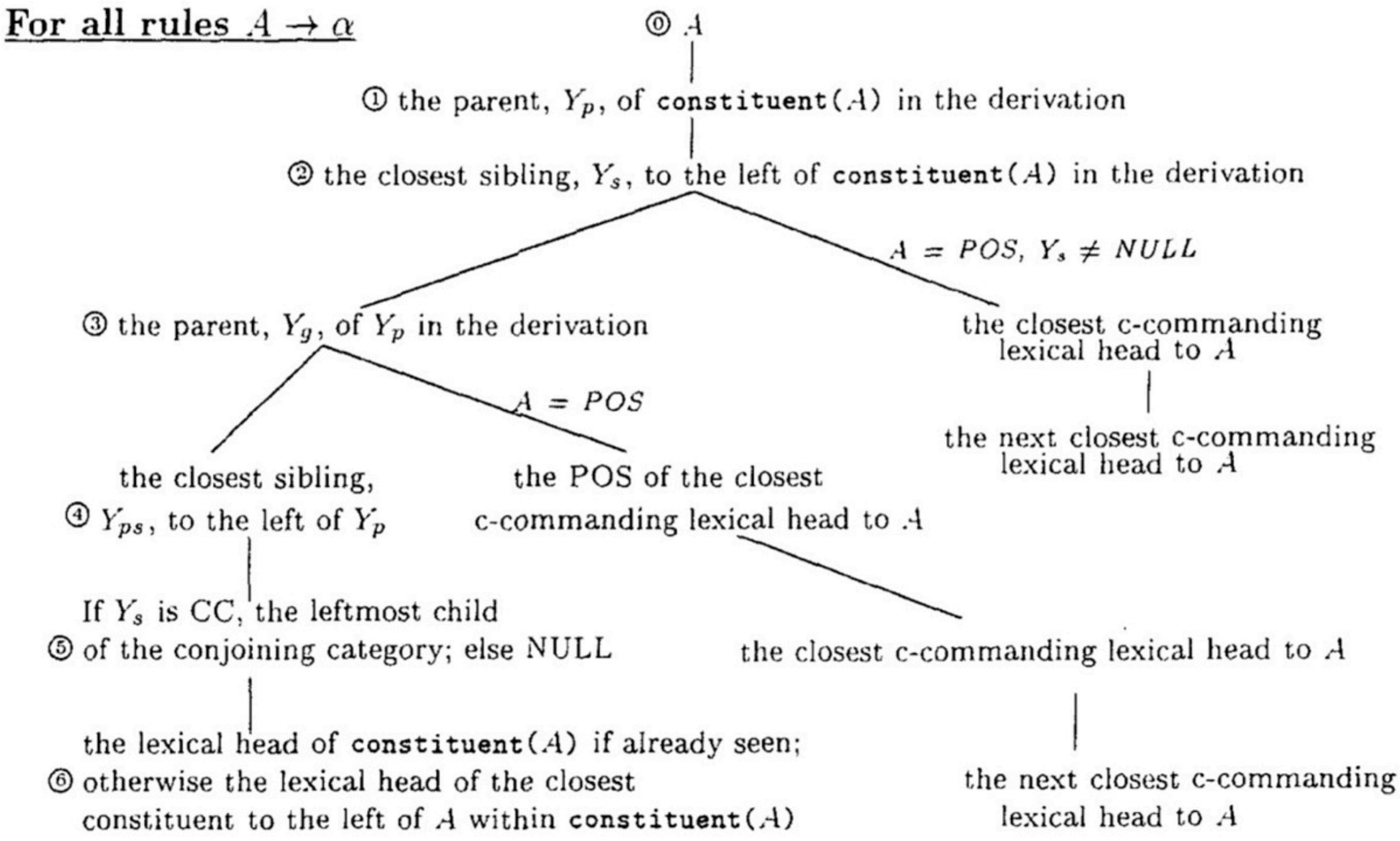
c-command

Node A c-commands node B iff:

- A does not dominate B
- B does not dominate A
- The first branching node that dominates A, also dominates B



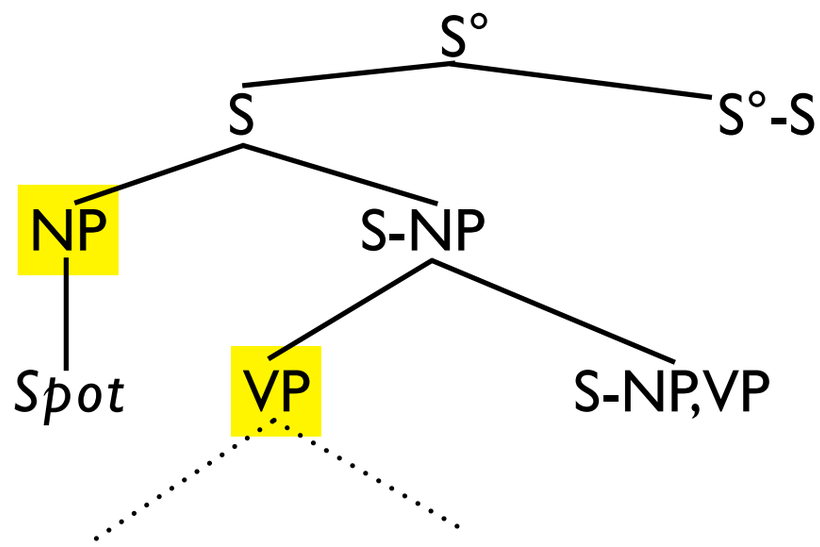
Conditioning



Conditioning

non-POS non-terminals *A*:

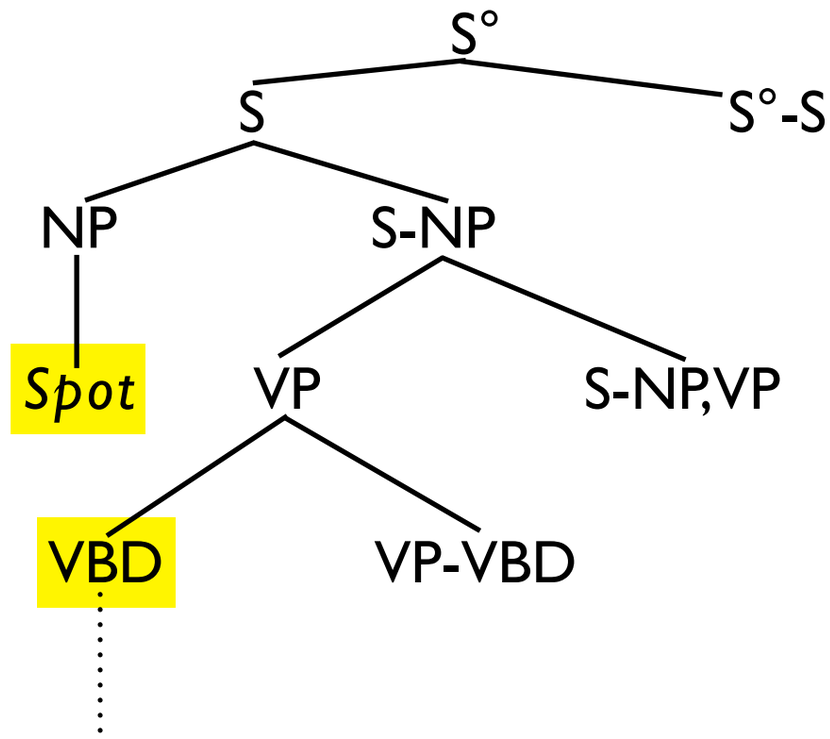
④ the closest sibling, Y_{ps}, to the left of Y_p



Conditioning

POS non-terminals *A*:

⑤ the closest c-commanding lexical head to *A*



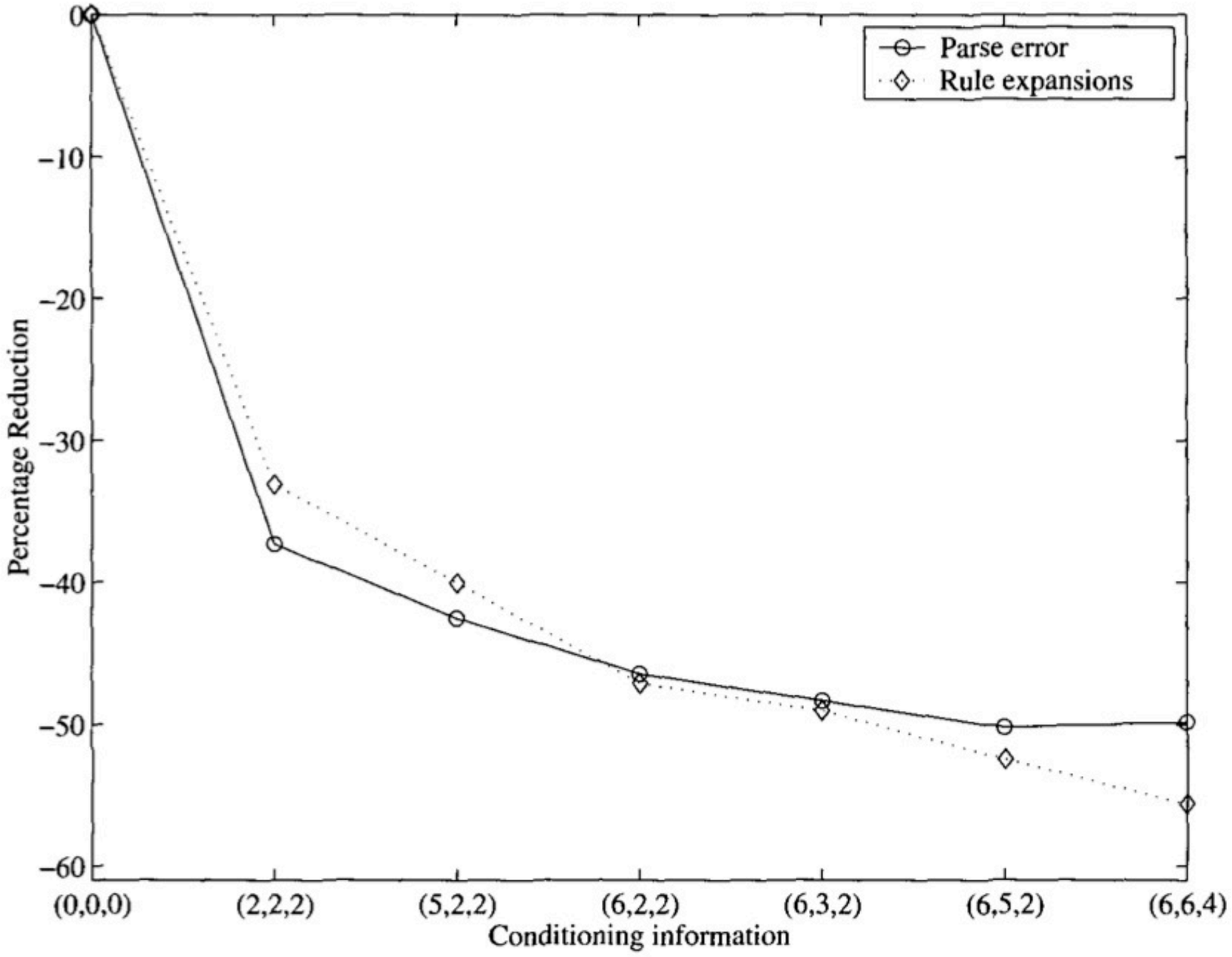
Conditioning

Conditioning Levels	Mnemonic Label	Information Level
(0,0,0)	none	simple PCFG
(2,2,2)	par+sib	small amount of structural context
(5,2,2)	NT struct	all structural context for non-POS
(6,2,2)	NT head	lexical context for non-POS
(6,3,2)	POS struct	more structural context for leftmost POS
(6,5,2)	attach	all attachment context for leftmost POS
(6,6,4)	all	everything

Evaluation

Conditioning	Labeled Recall	Labeled Precision	Percent Failed
none	71,1	75,3	0,9
par+sib	82,8	83,6	1,1
NT struct	84,3	84,9	1,0
NT head	85,6	85,7	0,9
POS struct	86,1	86,2	1,0
attach	86,7	86,6	1,2
all	86,6	86,5	1,3

Evaluation


Parse error
Rule expansions
0
−10
−20
−30
−40
−50
−60
Percentage Reduction
(0,0,0)
(2,2,2)
(5,2,2)
(6,2,2)
(6,3,2)
(6,5,2)
(6,6,4)
Conditioning information

Speech Recognition

Model	LM Weight	Word Error Rate	Sentence Error Rate
Roark	15	15,1	73,2
Treebank Trigram	5	16,5	79,8
no Language Model	0	16,8	84,0

Speech Recognition

- an incremental top-down parser can be used in a real-world application
- less accurate than state-of-the-art speech recognition systems
- results of interpolation with a trigram model imply that types of information are orthogonal

Summary

- probabilistic grammars and transform
- incremental top-down parsing
- conditioning based on already observed input and its structure
- application to speech recognition

Discussion

- benefits of grammar transforms
- improvement of conditioning
  - finer distinctions
- a parsing model as a language model
  - or as one of multiple models

Thank you for your attention.

Questions?!