Natural Language Processing
(A brief look at CKY, and then) Dependency parsing

Sofia Serrano
sofias6@cs.washington.edu

Credit to Yulia Tsvetkov for slides

Announcement(s)

- A3 is out on gitlab!

The CKY algorithm for parsing with PCFGs

Parsing

- Parsing is search through the space of all possible parses
  - e.g., we may want either any parse, all parses or the highest scoring parse (if PCFG):

$$\arg\max_{T \in G(x)} P(T)$$

- Bottom-up:
  - One starts from words and attempt to construct the full tree
- Top-down
  - Start from the start symbol and attempt to expand to get the sentence

CKY algorithm (aka CYK)

- Cocke-Kasami-Younger algorithm
  - Independently discovered in late 60s / early 70s

- An efficient bottom up parsing algorithm for (P)CFGs
  - can be used both for the recognition and parsing problems
  - Very important in NLP (and beyond)

- We will start with the non-probabilistic version

Constraints on the grammar

- The basic CKY algorithm supports only rules in the Chomsky Normal Form (CNF):

$$C \rightarrow x$$

Unary preterminal rules (generation of words given PoS tags)

$N \rightarrow telescope$ $\quad D \rightarrow the$

$$C \rightarrow C_1 C_2$$

Binary inner rules $S \rightarrow NP\ VP$ $\quad NP \rightarrow D\ N$

Constraints on the grammar

- The basic CKY algorithm supports only rules in the Chomsky Normal Form (CNF):

$$C \to x$$

$$C \to C_1 C_2$$

- Any CFG can be converted to an equivalent CNF
    - Equivalent means that they define the same language
    - However (syntactic) trees will look differently
    - It is possible to address it by defining such transformations that allows for easy reverse transformation

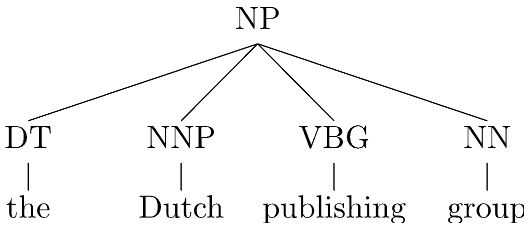
Transformation to CNF form

- What one need to do to convert to CNF form
    - Get rid of rules that mix terminals and non-terminals
    - Get rid of unary rules: $C \rightarrow C_1$
    - Get rid of N-ary rules: $C \rightarrow C_1\ C_2 \dots C_n \quad (n > 2)$

Crucial to process them, as required for efficient parsing

Transformation to CNF form: binarization

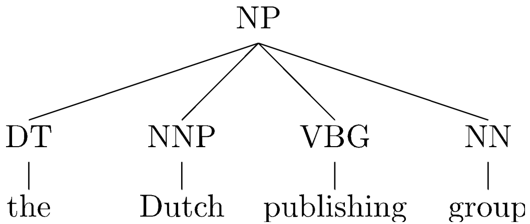
- Consider $NP \rightarrow DT \;\; NNP \;\; VBG \;\; NN$



- How do we get a set of binary rules which are equivalent?

Transformation to CNF form: binarization

- Consider $$NP \rightarrow DT \;\; NNP \;\; VBG \;\; NN$$

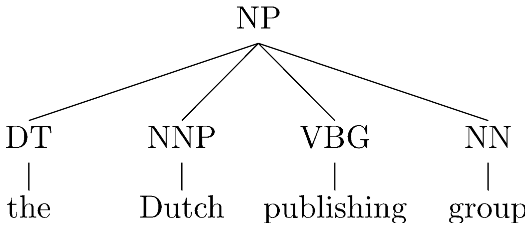


- How do we get a set of binary rules which are equivalent?

$$NP \rightarrow DT \;\; X$$
$$X \rightarrow NNP \;\; Y$$
$$Y \rightarrow VBG \;\; NN$$

Transformation to CNF form: binarization

- Consider $NP \rightarrow DT\ \ NNP\ \ VBG\ \ NN$



- How do we get a set of binary rules which are equivalent?

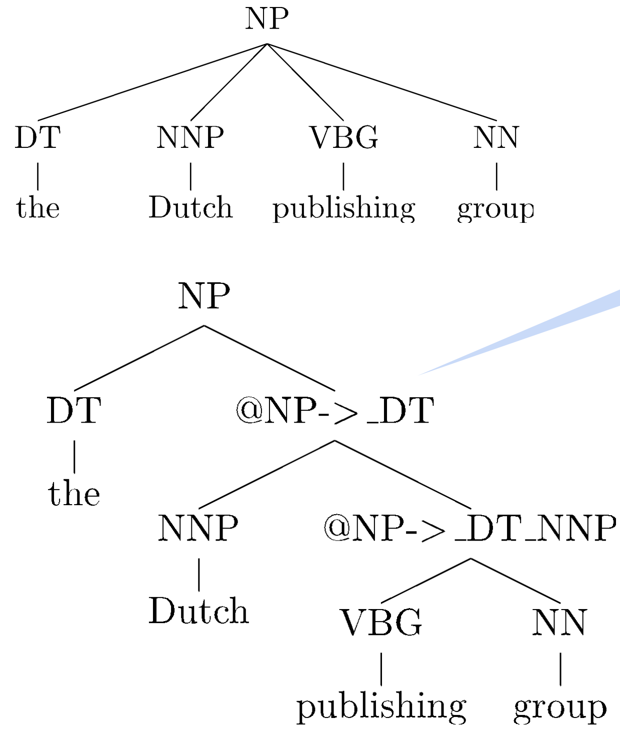
$$NP \rightarrow DT\ \ X$$
$$X \rightarrow NNP\ \ Y$$
$$Y \rightarrow VBG\ \ NN$$

- A more systematic way to refer to new non-terminals

$$NP \rightarrow DT\ \ @NP|DT$$
$$@NP|DT \rightarrow NNP\ \ @NP|DT\_NNP$$
$$@NP|DT\_NNP \rightarrow VBG\ \ NN$$

Transformation to CNF form: binarization

- Instead of binarizing tuples we can binarize trees on preprocessing:



Also known as **lossless Markovization** in the context of PCFGs

Can be easily reversed on postprocessing

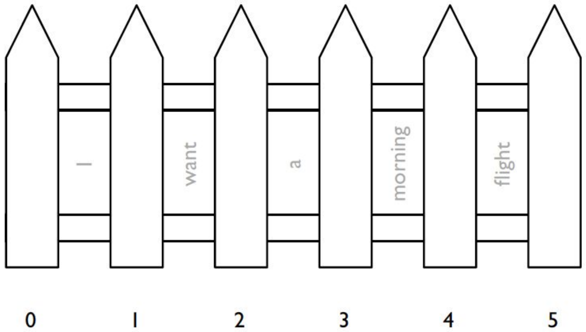
CKY: Parsing task

- We are given
  - a grammar <N, T, S, R>
  - a sequence of words $\boldsymbol{w} = (w_1, w_2, \ldots, w_n)$

- Our goal is to produce a parse tree for w

CKY: Parsing task

- We a given
  - a grammar <N, T, S, R>
  - a sequence of words $$\boldsymbol{w} = (w_1, w_2, \ldots, w_n)$$
- Our goal is to produce a parse tree for w
- We need an easy way to refer to substrings of w



indices refer to fenceposts

span (i, j) refers to words between fenceposts i and j

Parsing one word

$$C \rightarrow w_i$$

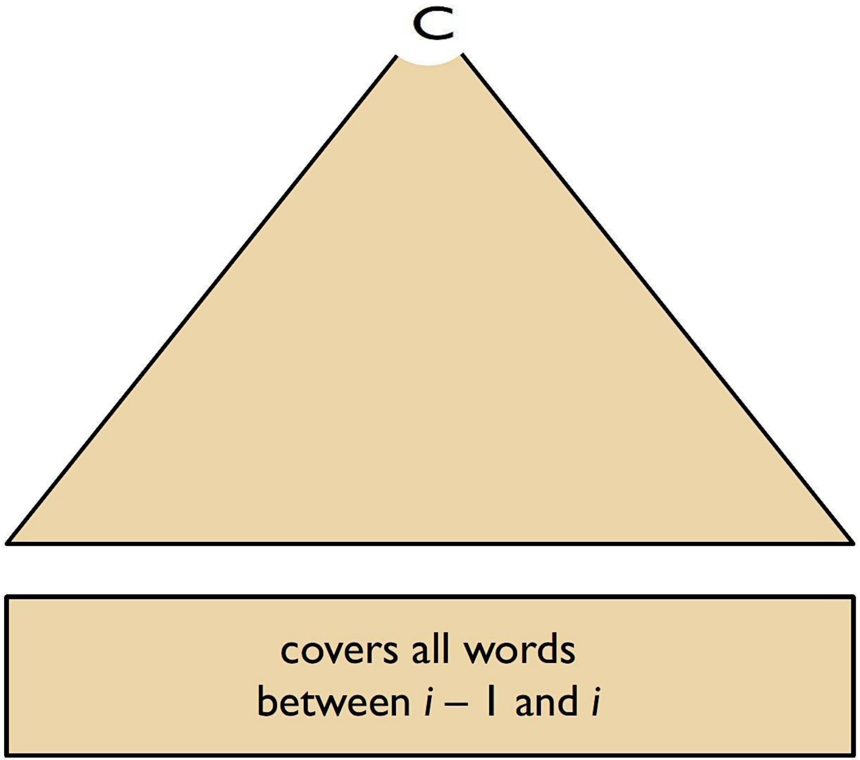
w_i

Parsing one word

$$C \rightarrow w_i$$

Parsing one word

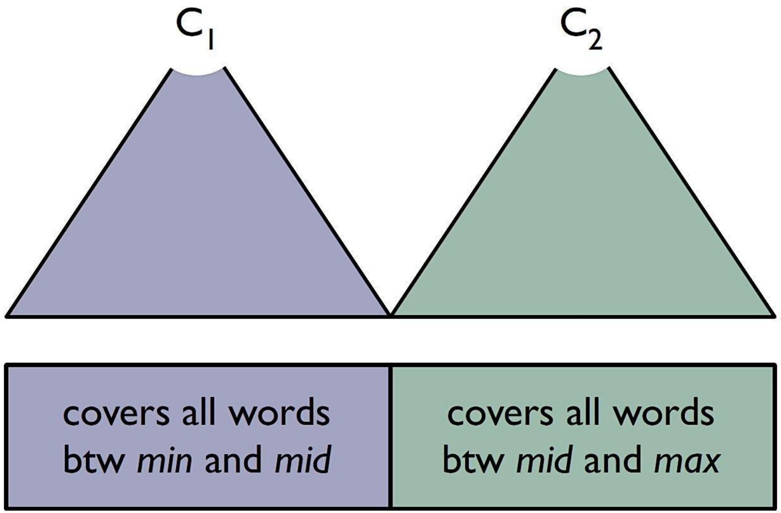
$$C \to w_i$$



Parsing longer spans

$$C \rightarrow C_1 \;\; C_2$$

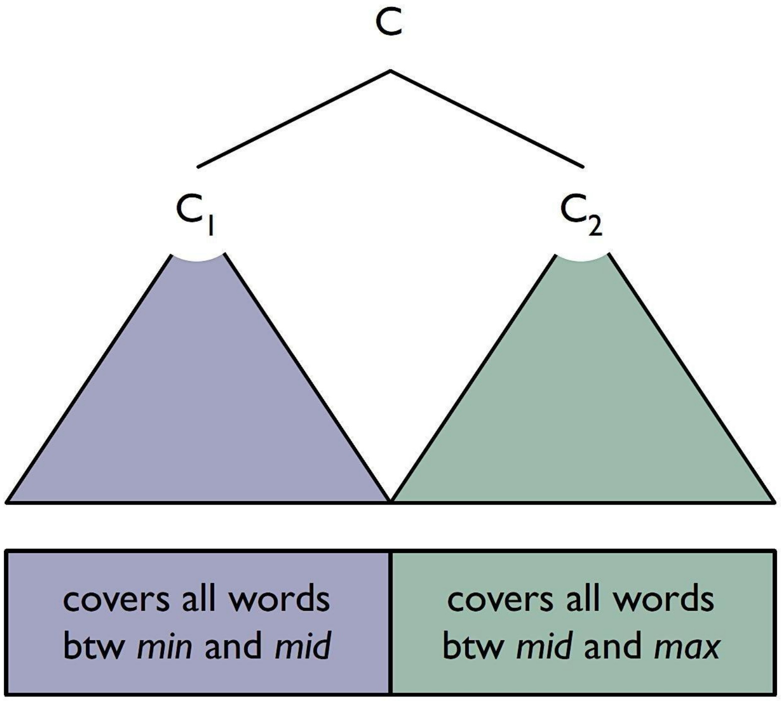
Check through all
C1, C2, mid



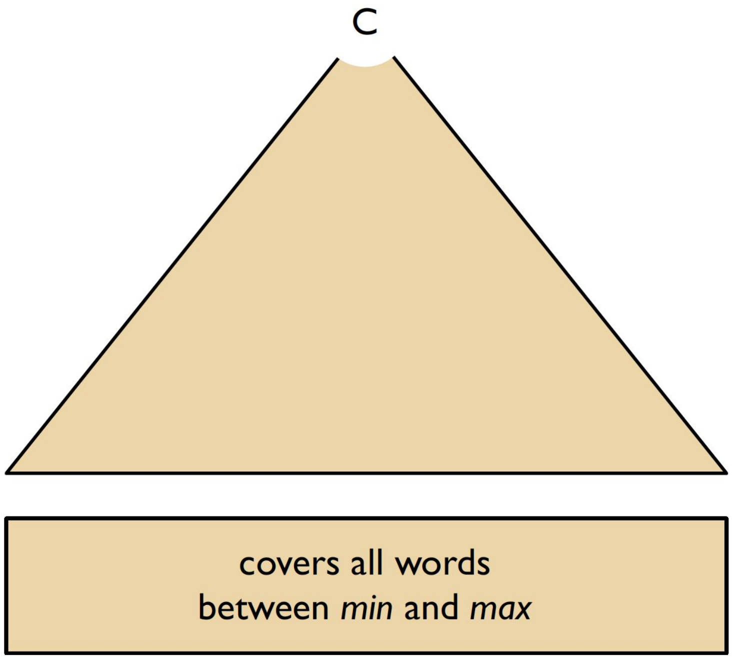
Parsing longer spans

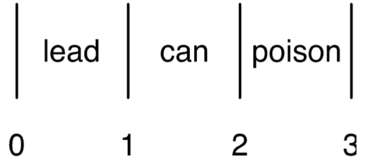
$C \rightarrow C_1 \;\; C_2$

Check through all C1, C2, mid



Parsing longer spans





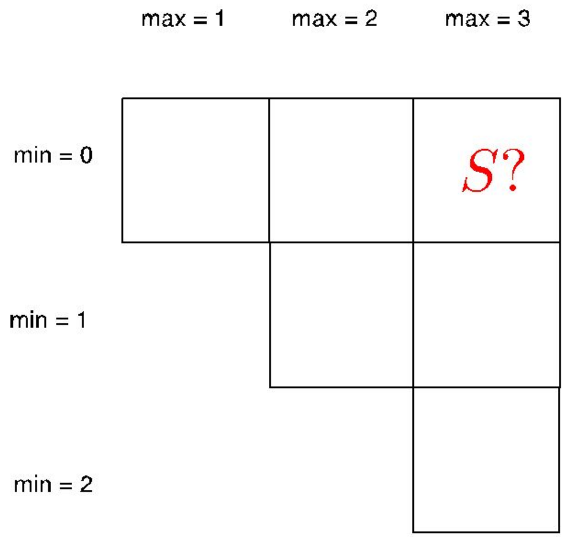


Chart (aka parsing triangle)

Inner rules

$$S \rightarrow NP\ VP$$

$$VP \rightarrow M\ V$$
$$VP \rightarrow V$$

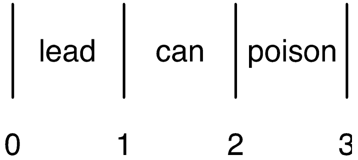
$$NP \rightarrow N$$
$$NP \rightarrow N\ NP$$

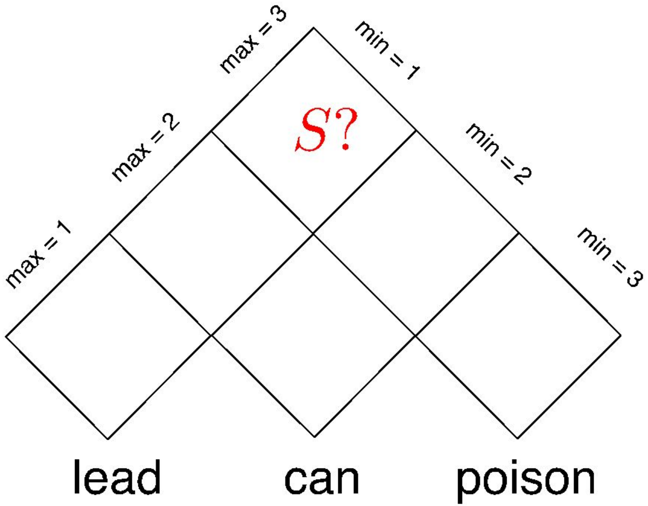
Preterminal rules

$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$

$$M \rightarrow can$$
$$M \rightarrow must$$

$$V \rightarrow poison$$
$$V \rightarrow lead$$





$$S \to NP\ VP$$

$$VP \to M\ V$$
$$VP \to V$$

$$NP \to N$$
$$NP \to N\ NP$$

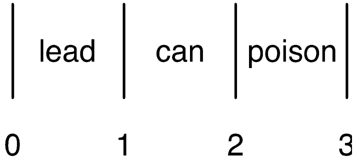
Inner rules

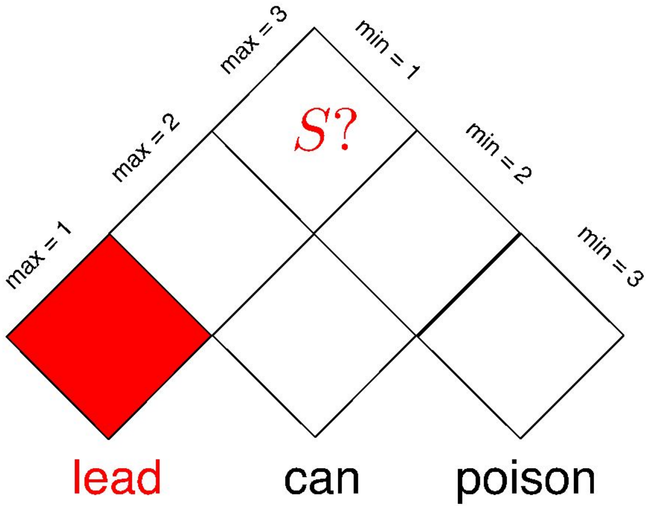
$$N \to can$$
$$N \to lead$$
$$N \to poison$$

$$M \to can$$
$$M \to must$$

$$V \to poison$$
$$V \to lead$$

Preterminal rules





$$S \rightarrow NP \ VP$$

$$VP \rightarrow M \ V$$
$$VP \rightarrow V$$

$$NP \rightarrow N$$
$$NP \rightarrow N \ NP$$

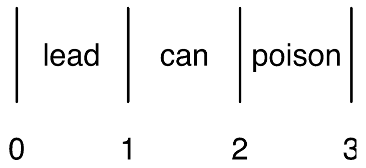
Inner rules

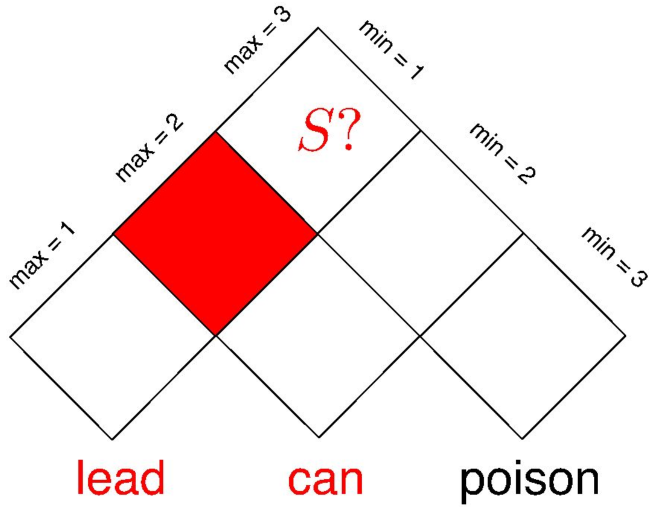
$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$

$$M \rightarrow can$$
$$M \rightarrow must$$

$$V \rightarrow poison$$
$$V \rightarrow lead$$

Preterminal rules





$$S \to NP\ \ VP$$

$$VP \to M\ \ V$$
$$VP \to V$$

$$NP \to N$$
$$NP \to N\ NP$$

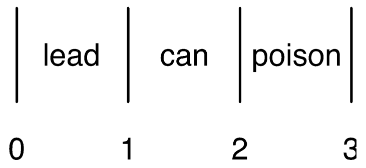
Inner rules

$$N \to can$$
$$N \to lead$$
$$N \to poison$$

$$M \to can$$
$$M \to must$$

$$V \to poison$$
$$V \to lead$$

Preterminal rules



	max = 1	max = 2	max = 3
min = 0			$S?$
min = 1			
min = 2			

Inner rules

$$S \rightarrow NP\ VP$$

$$VP \rightarrow M\ V$$

$$VP \rightarrow V$$

$$NP \rightarrow N$$

$$NP \rightarrow N\ NP$$

Preterminal rules

$$N \rightarrow can$$

$$N \rightarrow lead$$

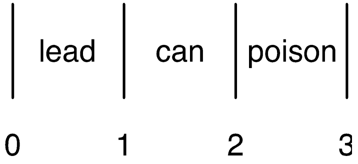
$$N \rightarrow poison$$

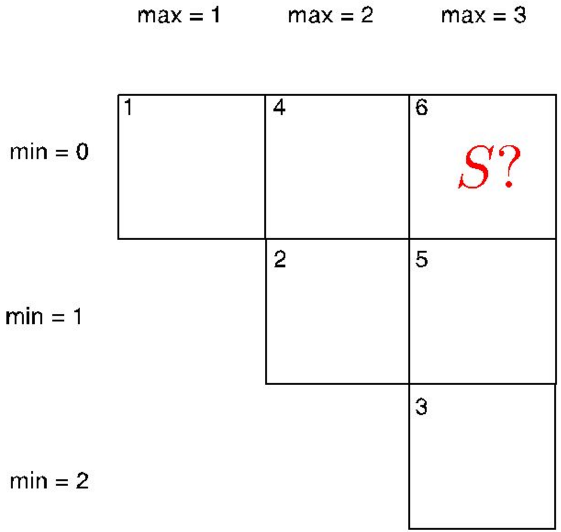
$$M \rightarrow can$$

$$M \rightarrow must$$

$$V \rightarrow poison$$

$$V \rightarrow lead$$





$$S \rightarrow NP\ VP$$

$$VP \rightarrow M\ V$$
$$VP \rightarrow V$$

$$NP \rightarrow N$$
$$NP \rightarrow N\ NP$$

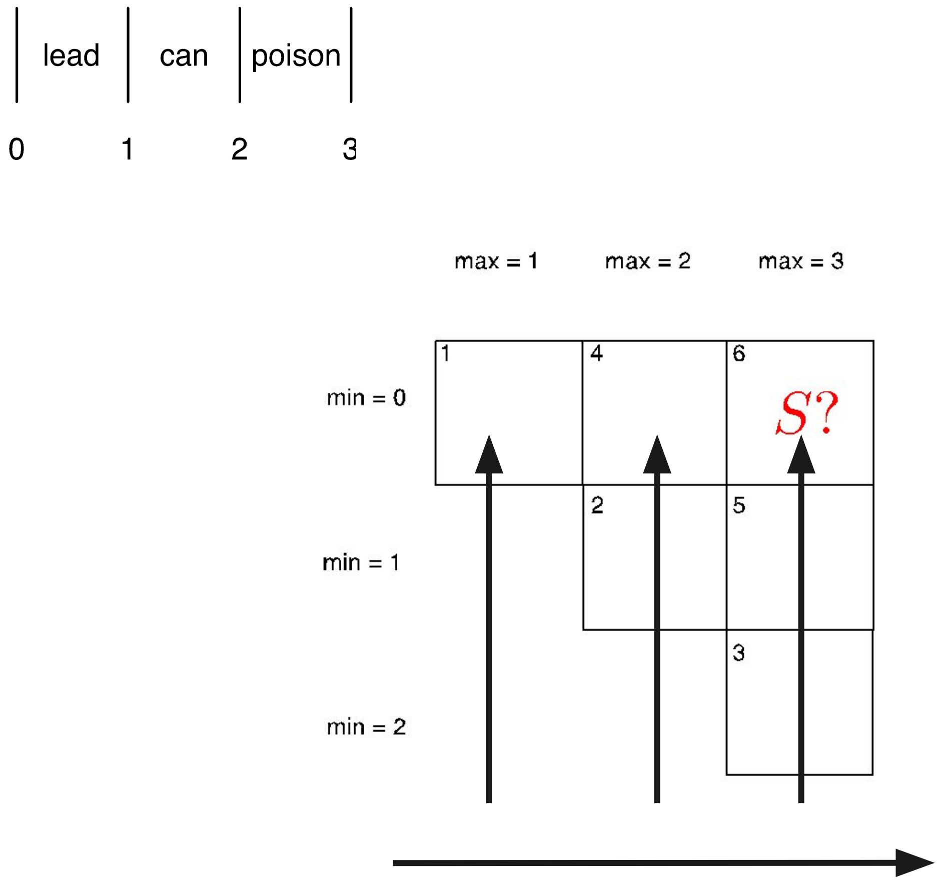
Inner rules

$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$

$$M \rightarrow can$$
$$M \rightarrow must$$

$$V \rightarrow poison$$
$$V \rightarrow lead$$

Preterminal rules



$$S \to NP\ VP$$

$$VP \to M\ V$$
$$VP \to V$$

$$NP \to N$$
$$NP \to N\ NP$$

Inner rules

$$N \to can$$
$$N \to lead$$
$$N \to poison$$

$$M \to can$$
$$M \to must$$

$$V \to poison$$
$$V \to lead$$

Preterminal rules

lead	can	poison

0 1 2 3

	max = 1	max = 2	max = 3
min = 0	1 ?		
min = 1		2 ?	
min = 2			3 ?

$$S \rightarrow NP \ VP$$

$$VP \rightarrow M \ V$$
$$VP \rightarrow V$$

$$NP \rightarrow N$$
$$NP \rightarrow N \ NP$$

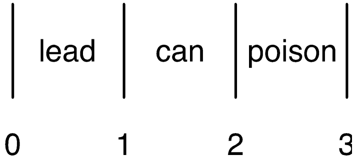
Inner rules

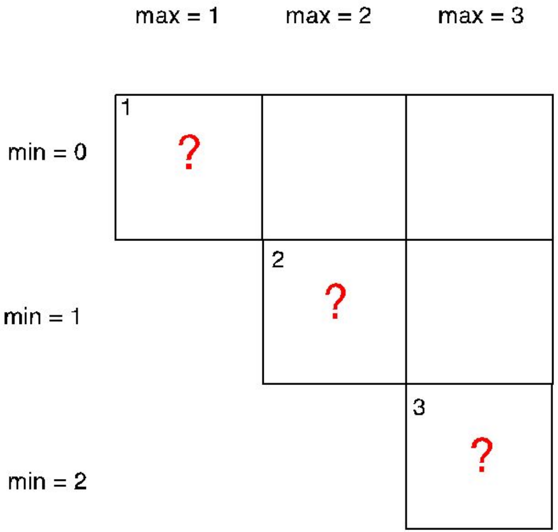
$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$

$$M \rightarrow can$$
$$M \rightarrow must$$

$$V \rightarrow poison$$
$$V \rightarrow lead$$

Preterminal rules





$$S \rightarrow NP\ VP$$

$$VP \rightarrow M\ V$$
$$VP \rightarrow V$$

$$NP \rightarrow N$$
$$NP \rightarrow N\ NP$$

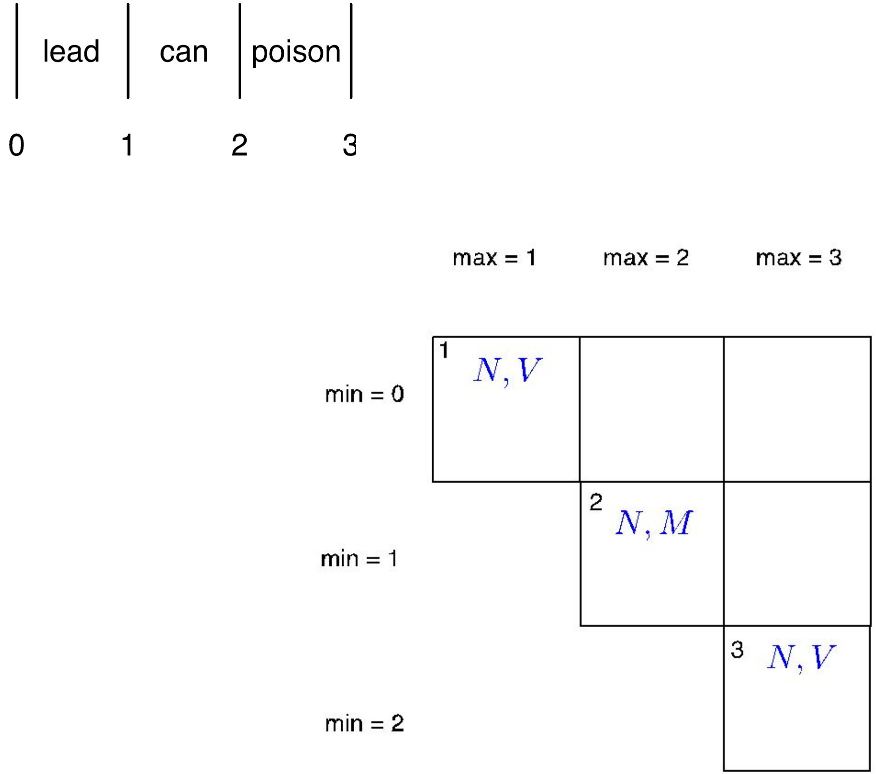
Inner rules

$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$

$$M \rightarrow can$$
$$M \rightarrow must$$

$$V \rightarrow poison$$
$$V \rightarrow lead$$

Preterminal rules



$$S \to NP\ VP$$

$$VP \to M\ V$$
$$VP \to V$$

$$NP \to N$$
$$NP \to N\ NP$$

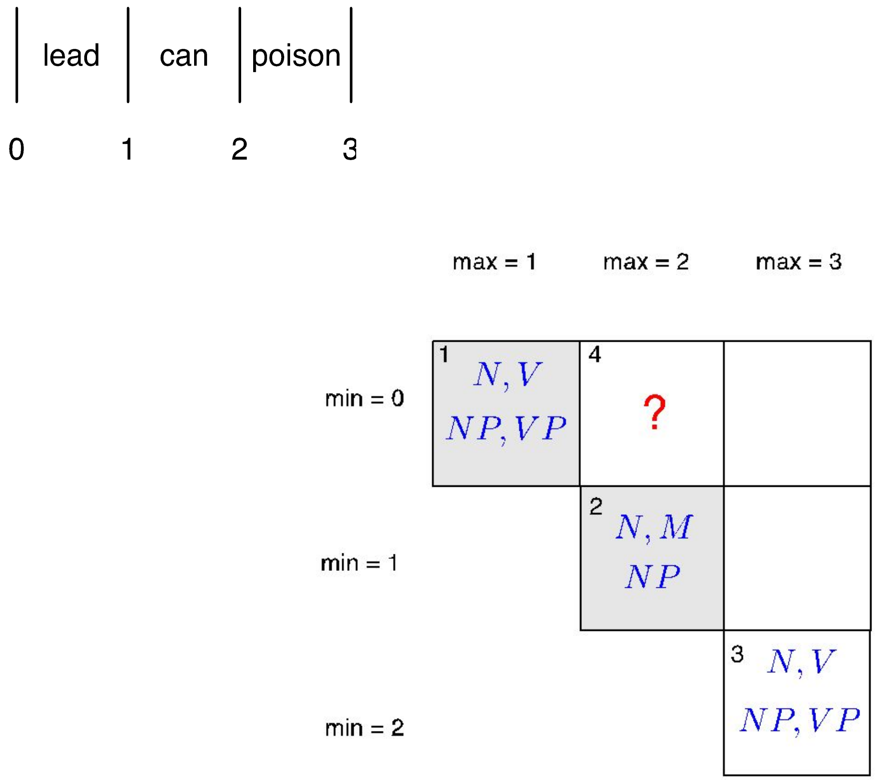
Inner rules

$$N \to can$$
$$N \to lead$$
$$N \to poison$$

$$M \to can$$
$$M \to must$$

$$V \to poison$$
$$V \to lead$$

Preterminal rules



$$S \rightarrow NP \ VP$$

$$VP \rightarrow M \ V$$
$$VP \rightarrow V$$

$$NP \rightarrow N$$
$$NP \rightarrow N \ NP$$

Inner rules

$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$

$$M \rightarrow can$$
$$M \rightarrow must$$

$$V \rightarrow poison$$
$$V \rightarrow lead$$

Preterminal rules

| lead | can | poison |

0 1 2 3

	max = 1	max = 2	max = 3
min = 0	1 N, V NP, VP	4 ?	
min = 1		2 N, M NP	
min = 2			3 N, V NP, VP

$$S \rightarrow NP\ VP$$

Inner rules

$$VP \rightarrow M\ V$$
$$VP \rightarrow V$$
$$NP \rightarrow N$$
$$\boxed{NP \rightarrow N\ NP}$$

Preterminal rules

$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$
$$M \rightarrow can$$
$$M \rightarrow must$$
$$V \rightarrow poison$$
$$V \rightarrow lead$$

	lead		can		poison	
0		1		2		3

	max = 1	max = 2	max = 3
min = 0	1 N, V NP, VP	4 NP	
min = 1		2 N, M NP	
min = 2			3 N, V NP, VP

$$S \rightarrow NP\ VP$$

$$VP \rightarrow M\ V$$
$$VP \rightarrow V$$

$$NP \rightarrow N$$
$$\boxed{NP \rightarrow N\ NP}$$

Inner rules

$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$

$$M \rightarrow can$$
$$M \rightarrow must$$

$$V \rightarrow poison$$
$$V \rightarrow lead$$

Preterminal rules

| lead | can | poison |

0 1 2 3

	max = 1	max = 2	max = 3
min = 0	1 N, V NP, VP	4 NP	
min = 1		2 N, M NP	5 ?
min = 2			3 N, V NP, VP

$$S \rightarrow NP \ VP$$

$$VP \rightarrow M \ V$$

$$VP \rightarrow V$$

$$NP \rightarrow N$$

$$NP \rightarrow N \ NP$$

Inner rules

$$N \rightarrow can$$

$$N \rightarrow lead$$

$$N \rightarrow poison$$

$$M \rightarrow can$$

$$M \rightarrow must$$

$$V \rightarrow poison$$

$$V \rightarrow lead$$

Preterminal rules

| lead | can | poison |

0 1 2 3

	max = 1	max = 2	max = 3
min = 0	1 N, V NP, VP	4 NP	
min = 1		2 N, M NP	5 $S, VP,$ NP
min = 2			3 N, V NP, VP

$\boxed{S \rightarrow NP\ VP}$

$\boxed{VP \rightarrow M\ V}$

$VP \rightarrow V$

$NP \rightarrow N$

$\boxed{NP \rightarrow N\ NP}$

Inner rules

$N \rightarrow can$

$N \rightarrow lead$

$N \rightarrow poison$

$M \rightarrow can$

$M \rightarrow must$

$V \rightarrow poison$

$V \rightarrow lead$

Preterminal rules

	lead		can		poison	
0		1		2		3

	max = 1	max = 2	max = 3
min = 0	1 N, V NP, VP	4 NP	6 ?
min = 1		2 N, M NP	5 $S, VP,$ NP
min = 2			3 N, V NP, VP

$$S \rightarrow NP \;\; VP$$

$$VP \rightarrow M \;\; V$$

$$VP \rightarrow V$$

$$NP \rightarrow N$$

$$NP \rightarrow N \; NP$$

Inner rules

$$N \rightarrow can$$

$$N \rightarrow lead$$

$$N \rightarrow poison$$

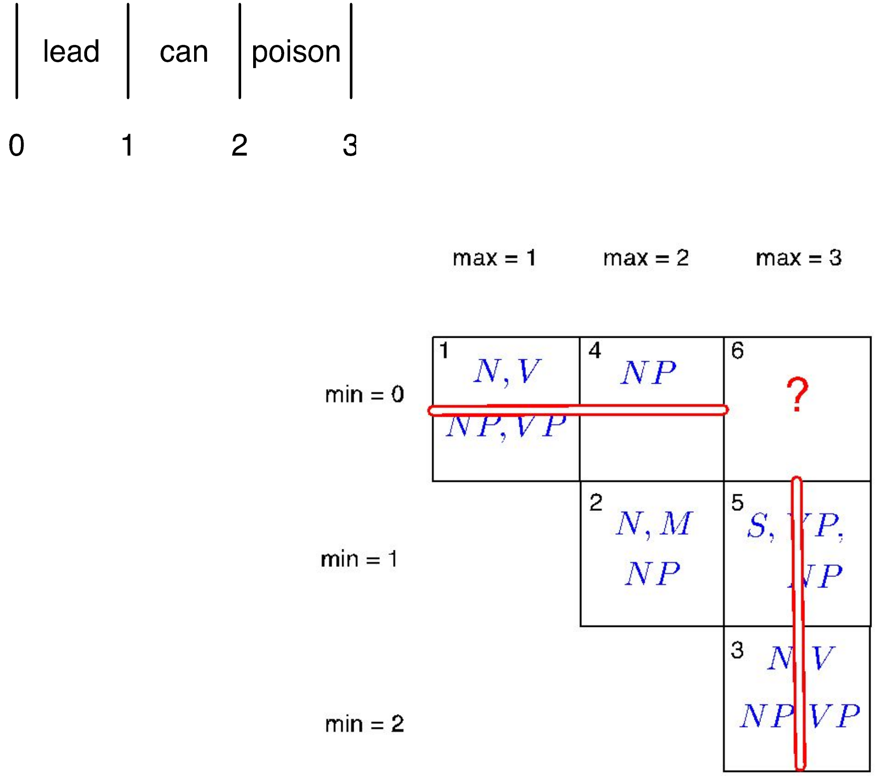
$$M \rightarrow can$$

$$M \rightarrow must$$

$$V \rightarrow poison$$

$$V \rightarrow lead$$

Preterminal rules



$$S \rightarrow NP\ VP$$

$$VP \rightarrow M\ V$$
$$VP \rightarrow V$$

$$NP \rightarrow N$$
$$NP \rightarrow N\ NP$$

Inner rules

$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$

$$M \rightarrow can$$
$$M \rightarrow must$$

$$V \rightarrow poison$$
$$V \rightarrow lead$$

Preterminal rules

0		1		2		3
	lead		can		poison	

mid=1

	max = 1	max = 2	max = 3
min = 0	1 N, V NP, VP	4 NP	6 S, NP
min = 1		2 N, M NP	5 $S, VP,$ NP
min = 2			3 N, V NP, VP

$$\boxed{S \rightarrow NP\ VP}$$

$$VP \rightarrow M\ \ V$$
$$VP \rightarrow V$$

$$NP \rightarrow N$$
$$\boxed{NP \rightarrow N\ NP}$$

Inner rules

$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$

$$M \rightarrow can$$
$$M \rightarrow must$$

$$V \rightarrow poison$$
$$V \rightarrow lead$$

Preterminal rules

$$S \rightarrow NP\ VP$$

| lead | can | poison |

0 1 2 3

mid=2

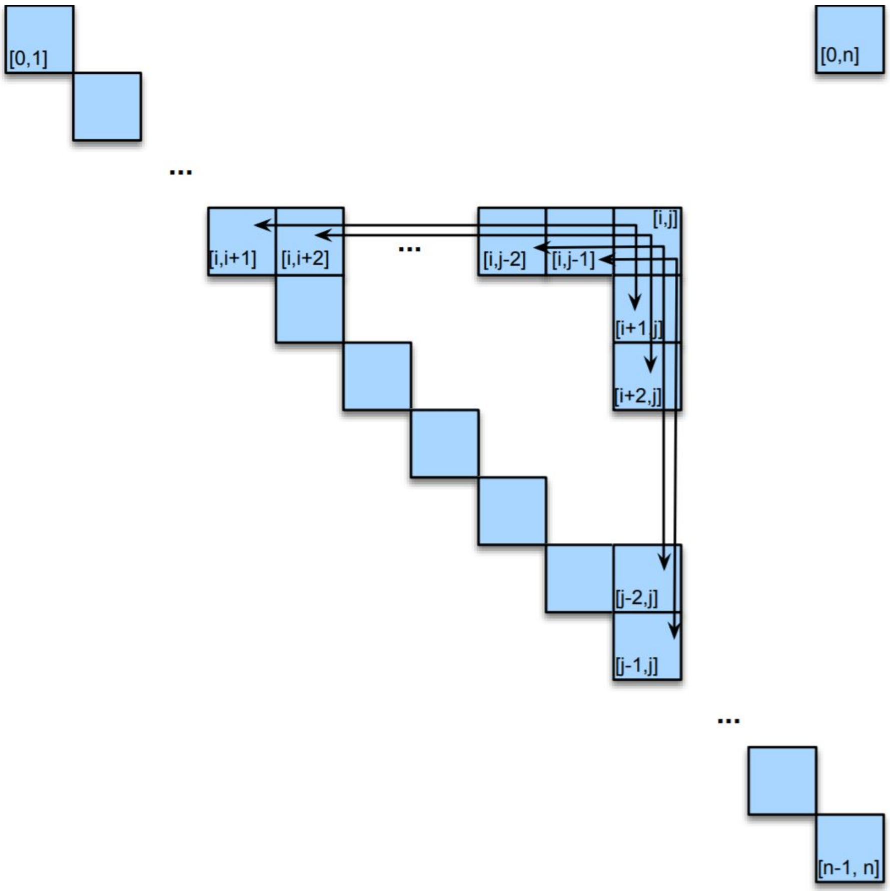
	max = 1	max = 2	max = 3
min = 0	1 N, V NP, VP	4 NP	6 S, NP $S(?!)$
min = 1		2 N, M NP	5 $S, VP,$ NP
min = 2			3 N, V NP, VP

$$VP \rightarrow M\ V$$
$$VP \rightarrow V$$
$$NP \rightarrow N$$
$$NP \rightarrow N\ NP$$

Inner rules

$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$
$$M \rightarrow can$$
$$M \rightarrow must$$
$$V \rightarrow poison$$
$$V \rightarrow lead$$

Preterminal rules


[0,1]
[0,n]
...
[i,j]
[i,i+1]
[i,i+2]
...
[i,j-2]
[i,j-1]
[i+1,j]
[i+2,j]
[j-2,j]
[j-1,j]
...
[n-1, n]

lead	can	poison

0 1 2 3

	max = 1	max = 2	max = 3
min = 0	1 N, V NP, VP	4 NP	6 S, NP $S(?!)$
min = 1		2 N, M NP	5 $S, VP,$ NP
min = 2			3 N, V NP, VP

Apparently the sentence is ambiguous for the grammar: (as the grammar overgenerates)

$$\boxed{S \rightarrow NP\ VP}$$

$$VP \rightarrow M\ \ V$$
$$VP \rightarrow V$$

$$NP \rightarrow N$$
$$NP \rightarrow N\ NP$$

Inner rules

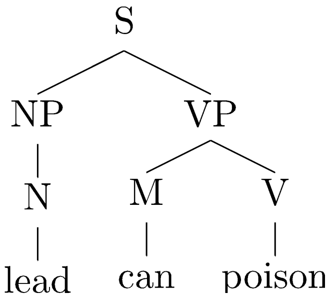
$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$

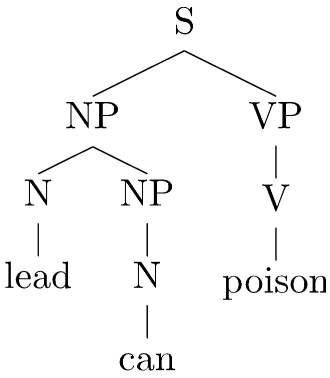
$$M \rightarrow can$$
$$M \rightarrow must$$

$$V \rightarrow poison$$
$$V \rightarrow lead$$

Preterminal rules

Ambiguity

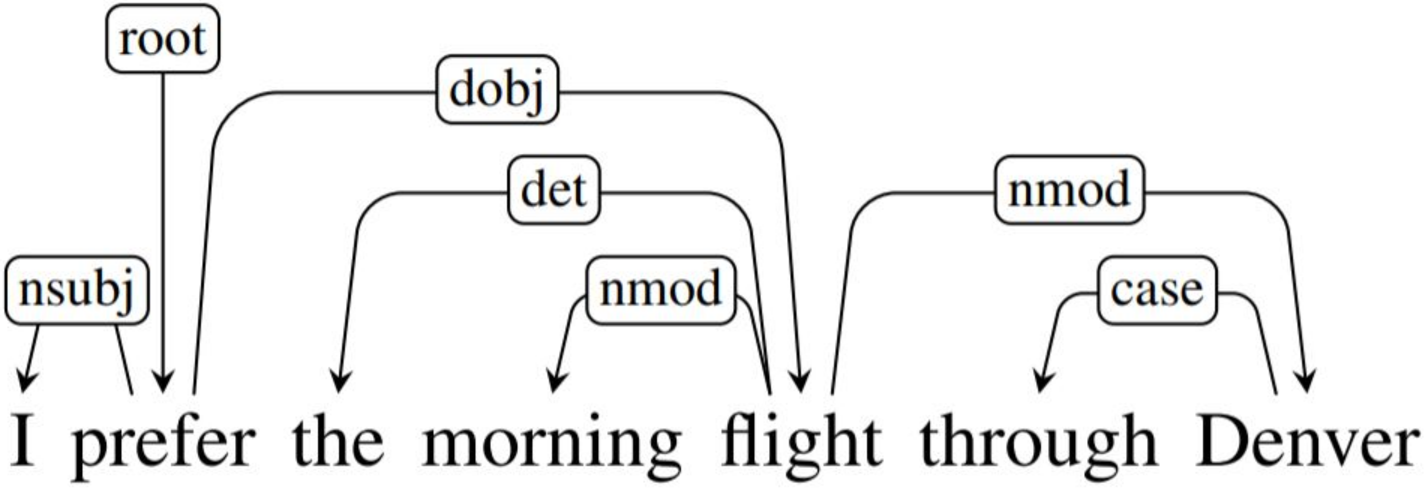




No subject-verb agreement, and *poison* used as an intransitive verb

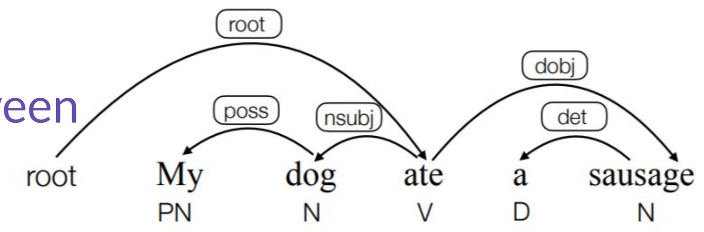
Dependency parsing

Dependency representation

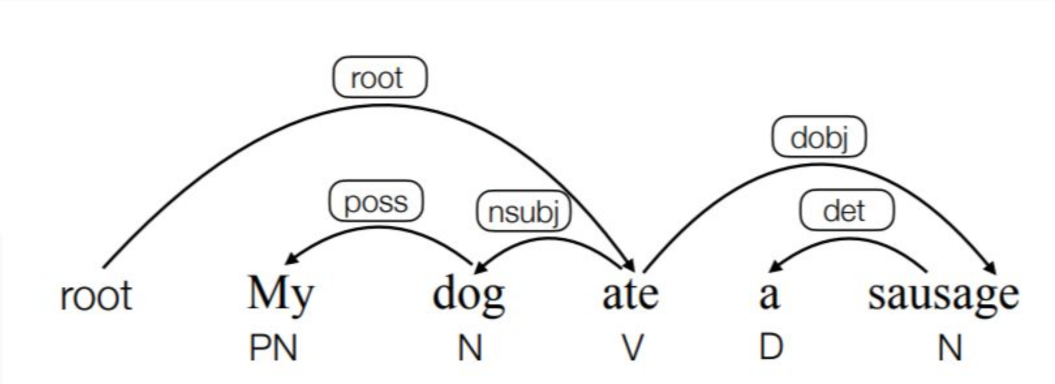


Dependency trees

- Nodes are words (along with part-of-speech tags)
- Directed arcs encode syntactic dependencies between them
- Labels are types of relations between the words
    - poss – possessive
    - dobj – direct object
    - nsub - subject
    - det - determiner

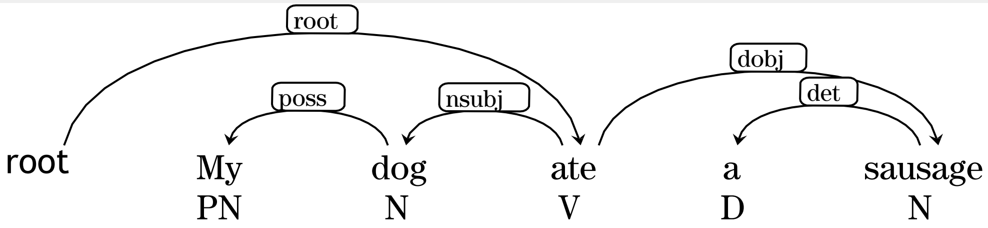


Recovering shallow semantics



- Some semantic information can be (approximately) derived from syntactic information
    - Subjects (nsubj) are (often) agents ("initiator / doers for an action")
    - Direct objects (dobj) are (often) patients ("affected entities")

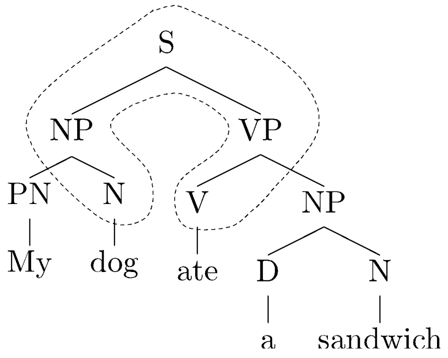
Recovering shallow semantics



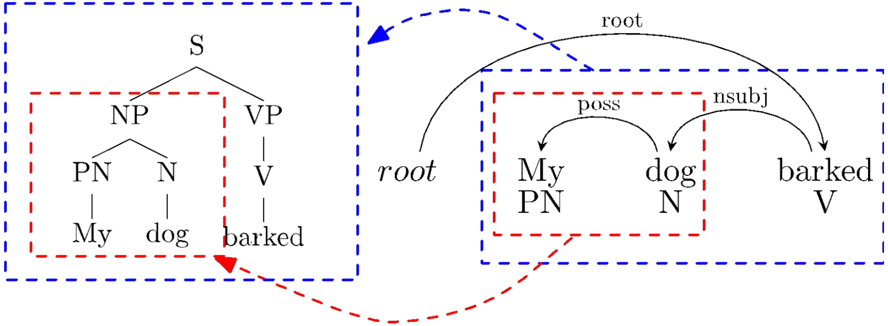
- Some semantic information can be (approximately) derived from syntactic information
  - Subjects (nsubj) are (often) agents ("initiator / doers for an action")
  - Direct objects (dobj) are (often) patients ("affected entities")
- But even for agents and patients consider:
  - Mary is baking a cake in the oven
  - A cake is baking in the oven
- In general it is not trivial even for the most shallow forms of semantics
  - E.g., consider prepositions: *in* can encode direction, position, temporal information, ...

Constituent and dependency representations

- Constituent trees can (potentially) be converted to dependency trees

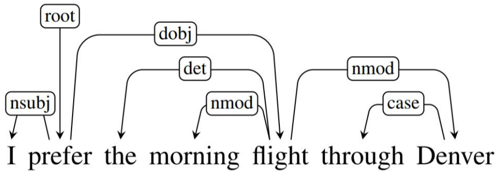


- Dependency trees can (potentially) be converted to constituent trees



Dependency representation

- A dependency structure can be defined as a directed graph G, consisting of
  - a set V of nodes – vertices, *words, punctuation, morphemes*
  - a set A of arcs – directed edges,
  - a linear precedence order < on V (word order).
- Labeled graphs
  - nodes in V are labeled with word forms (and annotation).
  - arcs in A are labeled with dependency types
  - $L = \{l_1, \ldots, l_{|L|}\}$ is the set of permissible arc labels;
  - Every arc in A is a triple (i,j,k), representing a dependency from w_i to w_j with label l_k.



Conversion from constituency to dependency

- Xia and Palmer (2001)
  - mark the head child of each node in a phrase structure, using the appropriate head rules
  - make the head of each non-head child depend on the head of the head-child

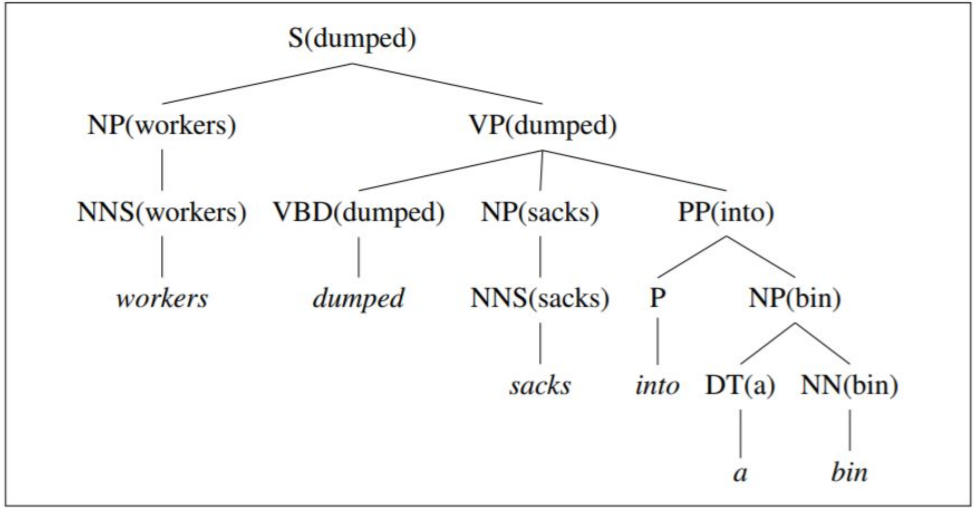


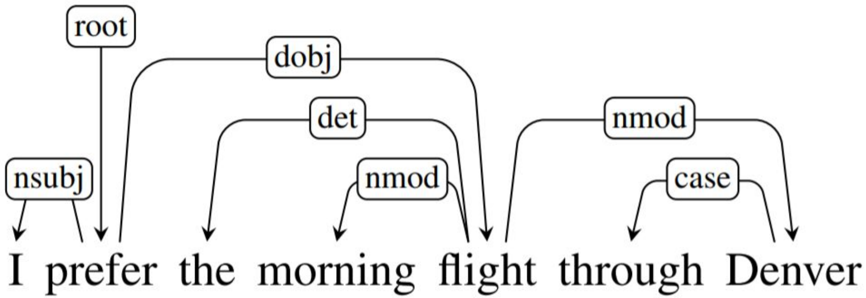
Figure 10.11 A lexicalized tree from Collins (1999).

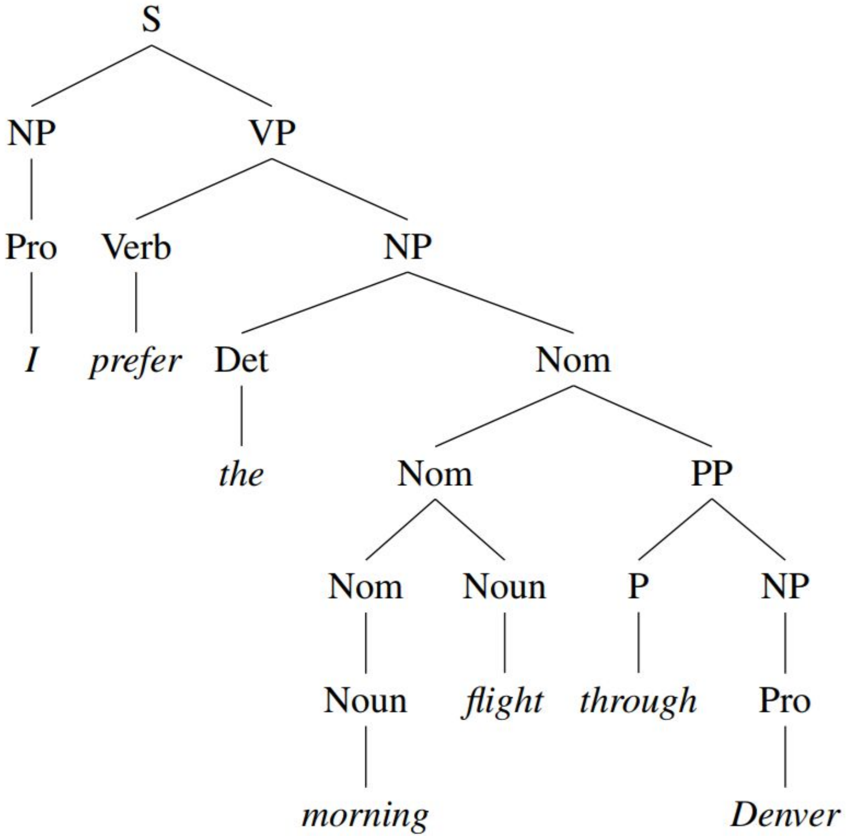
Dependency vs Constituency

- Dependency structures explicitly represent
  - head-dependent relations (directed arcs),
  - functional categories (arc labels)
  - possibly some structural categories (parts of speech)

- Phrase (aka constituent) structures explicitly represent
  - phrases (nonterminal nodes),
  - structural categories (nonterminal labels)

Dependency vs Constituency trees





Parsing Languages with Flexible Word Order

I prefer the morning flight through Denver

Я предпочитаю утренний перелет через Денвер

Languages with free word order

I prefer the morning flight through Denver

Я предпочитаю утренний перелет через Денвер

Я предпочитаю через Денвер утренний перелет

Утренний перелет я предпочитаю через Денвер

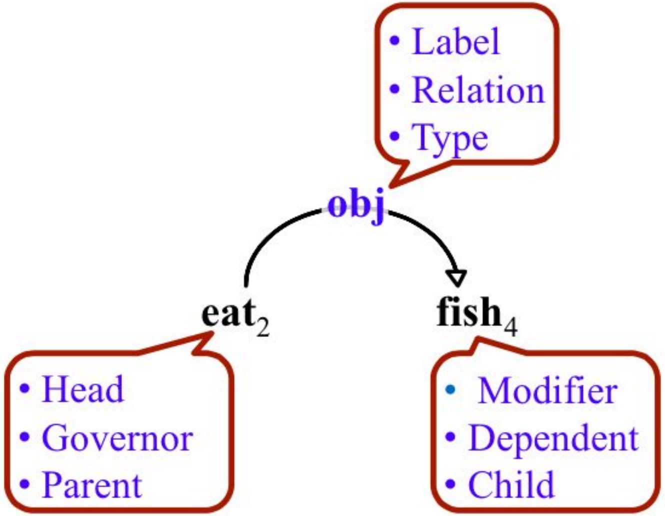
Перелет утренний я предпочитаю через Денвер

Через Денвер я предпочитаю утренний перелет

Я через Денвер предпочитаю утренний перелет

...

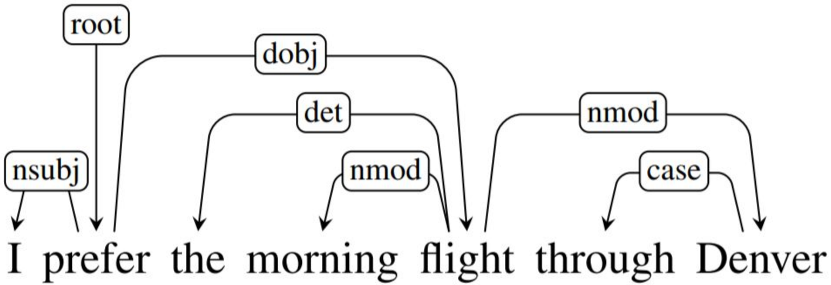
Dependency relations



Types of relationships

- The clausal relations NSUBJ and DOBJ identify the arguments: the subject and direct object of the predicate *cancel*

- The NMOD, DET, and CASE relations denote modifiers of the nouns *flights* and *Houston*.



Grammatical functions

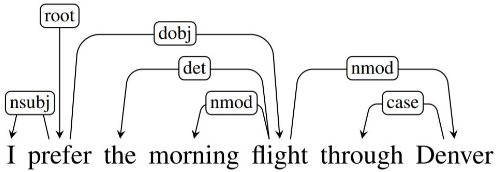
Clausal Argument Relations	Description
NSUBJ	Nominal subject
DOBJ	Direct object
IOBJ	Indirect object
CCOMP	Clausal complement
XCOMP	Open clausal complement
Nominal Modifier Relations	**Description**
NMOD	Nominal modifier
AMOD	Adjectival modifier
NUMMOD	Numeric modifier
APPOS	Appositional modifier
DET	Determiner
CASE	Prepositions, postpositions and other case markers
Other Notable Relations	**Description**
CONJ	Conjunct
CC	Coordinating conjunction

Figure 13.2 Selected dependency relations from the Universal Dependency set. (de Marneffe et al., 2014)

Dependency Constraints

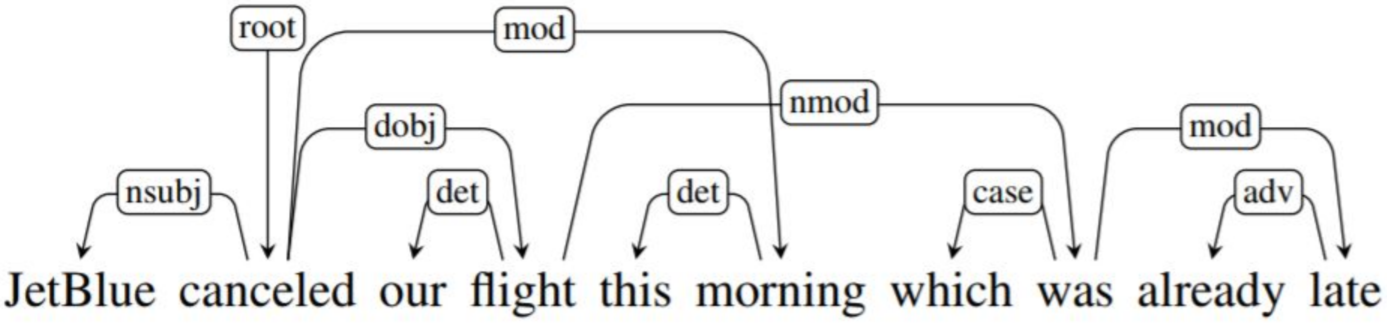
- Syntactic structure is complete (connectedness)
  - connectedness can be enforced by adding a special root node
- Syntactic structure is hierarchical (acyclicity)
  - there is a unique pass from the root to each vertex
- Every word has at most one syntactic head (single-head constraint)
  - except root that does not have incoming arcs

This makes the dependencies a tree



Projectivity

- Projective parse
  - arcs don't cross each other
  - mostly true for English
- Non-projective structures are needed to account for
  - long-distance dependencies
  - flexible word order



Projectivity

- Dependency grammars do not normally assume that all dependency-trees are projective, because some linguistic phenomena can only be achieved using non-projective trees.

- But a lot of parsers assume that the output trees are projective

- Reasons
  - conversion from constituency to dependency
  - the most widely used families of parsing algorithms impose projectivity

Non-Projective Statistics

Arabic: 11.2 %

Bulgarian: 5.4 %

Chinese: 0.0 %

Czech: 23.2 %

Danish: 15.6 %

Dutch: 36.4 %

German: 27.8 %

Japanese: 5.3 %

Polish: 18.9 %

Slovene: 22.2 %

Spanish 1.7 %

Swedish: 9.8 %

Turkish: 11.6 %

English: 0.0% (SD: 0.1%)

Percentage of non-projective trees for some treebanks of the CoNLL-X Shared Task and English.

Parsing problem

The parsing problem for a dependency parser is to find the optimal dependency tree $\boldsymbol{y}$ given an input sentence $\boldsymbol{x}$

This amounts to assigning a syntactic head $\boldsymbol{i}$

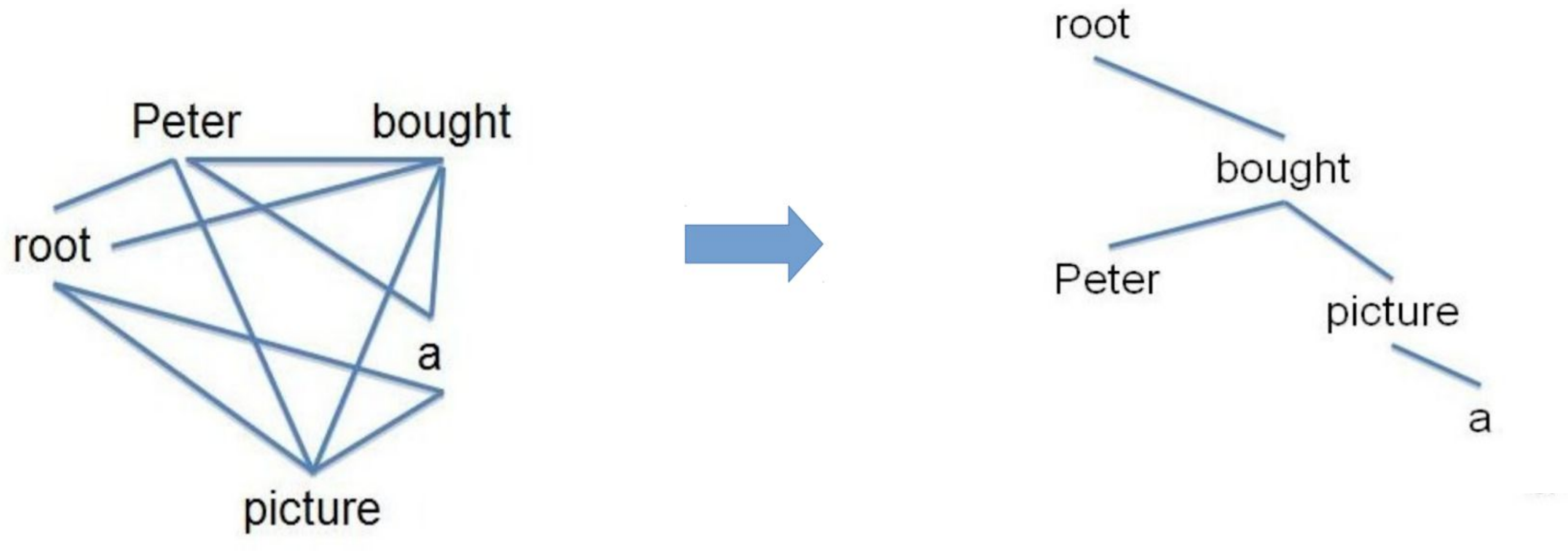
and a label $\boldsymbol{l}$ to every node $\boldsymbol{j}$ corresponding to a

word $\boldsymbol{x_j}$ in such a way that the resulting graph

is a tree rooted at the node 0

Parsing problem

- This is equivalent to finding a spanning tree in the complete graph containing all possible arcs

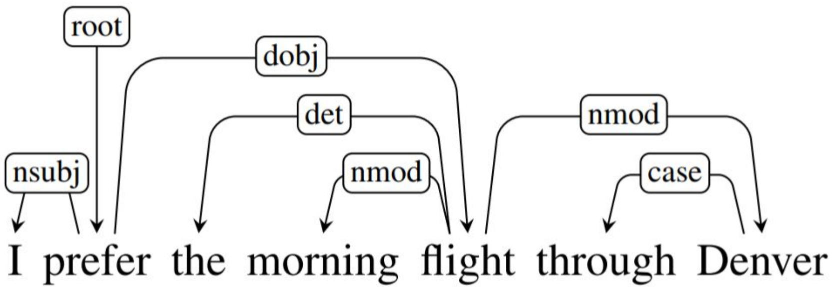


Parsing algorithms

- Transition based
  - greedy choice of local transitions guided by a good classifier
  - deterministic
  - MaltParser ([Nivre et al. 2008](#))
- Graph based
  - Minimum Spanning Tree for a sentence
  - McDonald et al.'s (2005) MSTParser
  - Martins et al.'s (2009) Turbo Parser

Transition Based Parsing

- greedy discriminative dependency parser
- motivated by a stack-based approach called **shift-reduce parsing** originally developed for analyzing programming languages (Aho & Ullman, 1972).
- Nivre 2003



Configuration

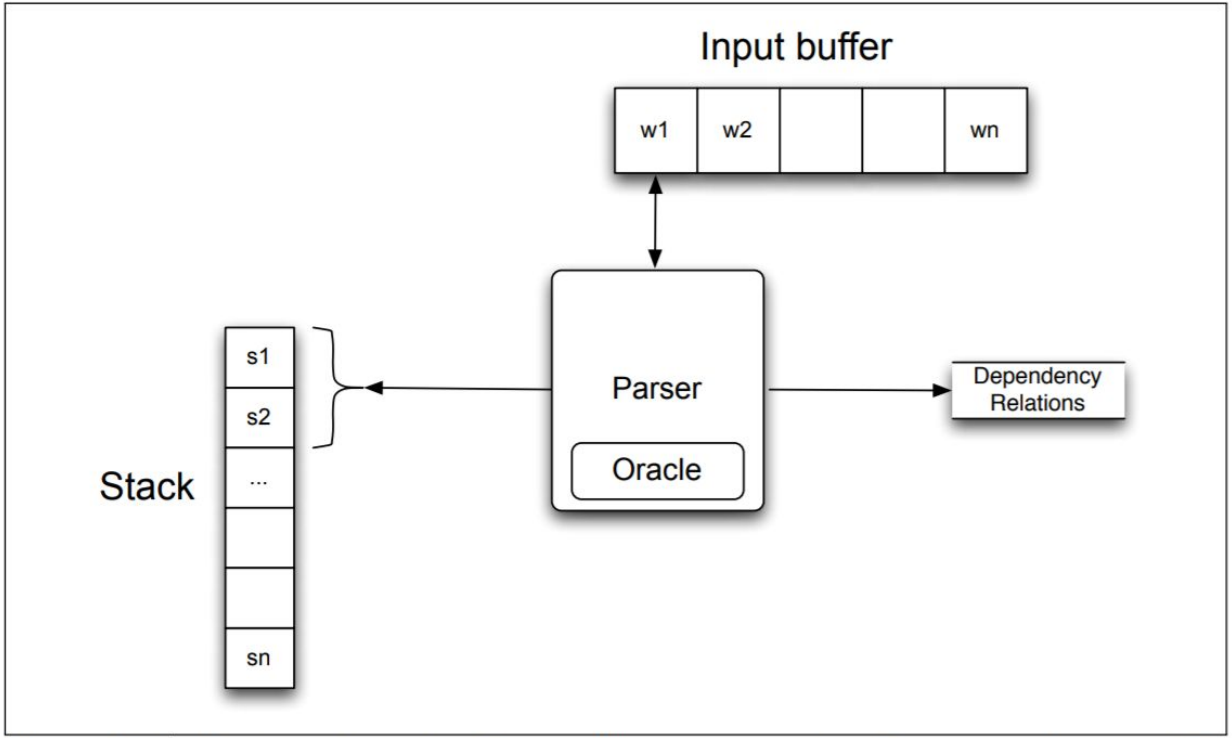


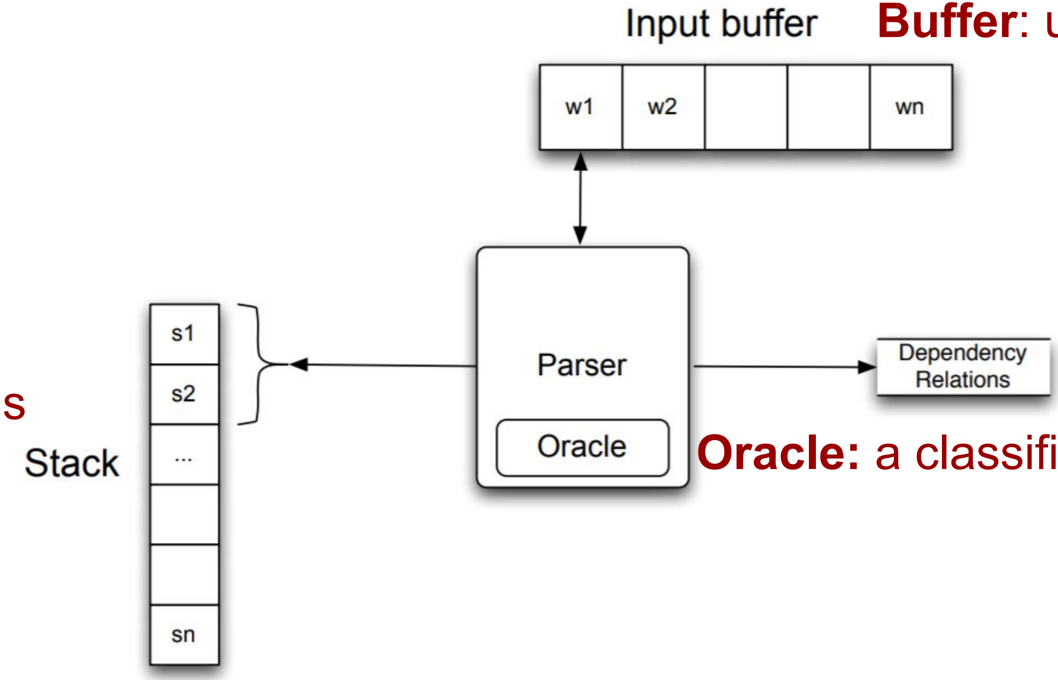
Figure 13.5 Basic transition-based parser. The parser examines the top two elements of the stack and selects an action based on consulting an oracle that examines the current configuration.

$$C = (\sigma, \beta, A)$$

Configuration

Buffer: unprocessed words

Stack: partially processed words



Oracle: a classifier

$$C_{\text{initial}} = ([\text{ROOT}], \boldsymbol{w}, \varnothing)$$

Operations

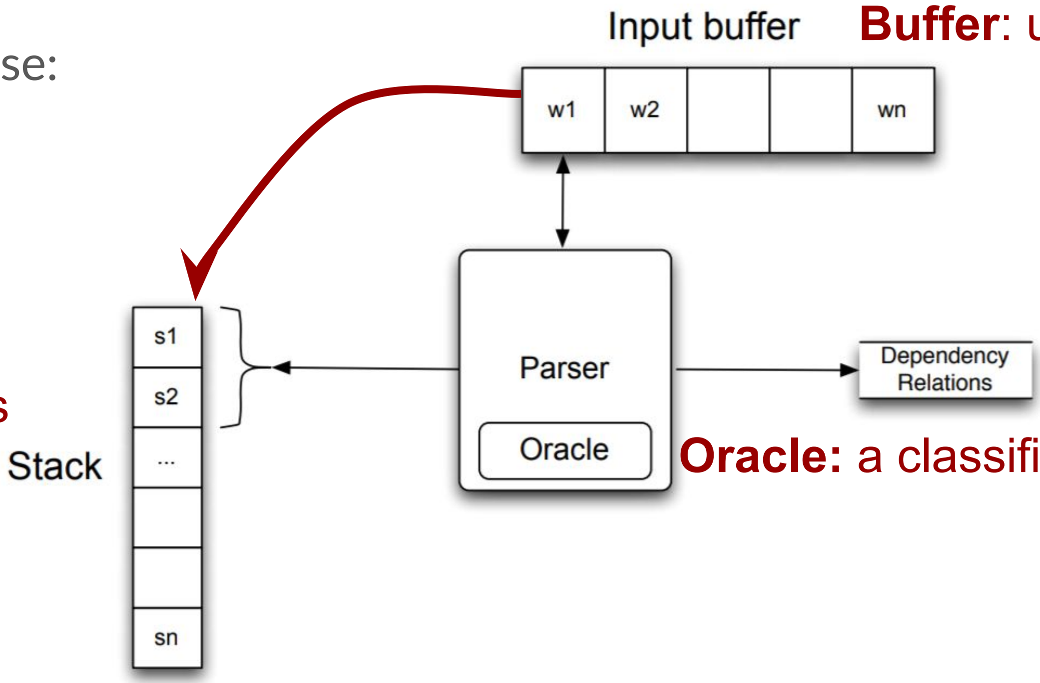
At each step choose:

- Shift

Buffer: unprocessed words

Stack: partially processed words

Oracle: a classifier



Operations

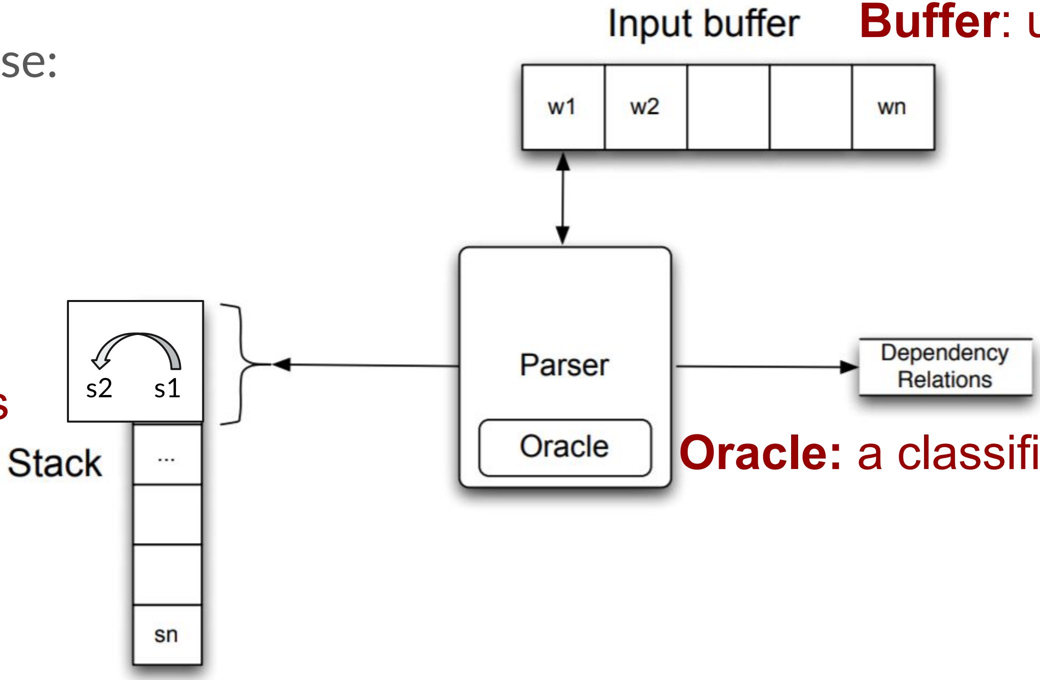
At each step choose:

- Shift
- Reduce left

Buffer: unprocessed words

Stack: partially processed words

Oracle: a classifier



Operations

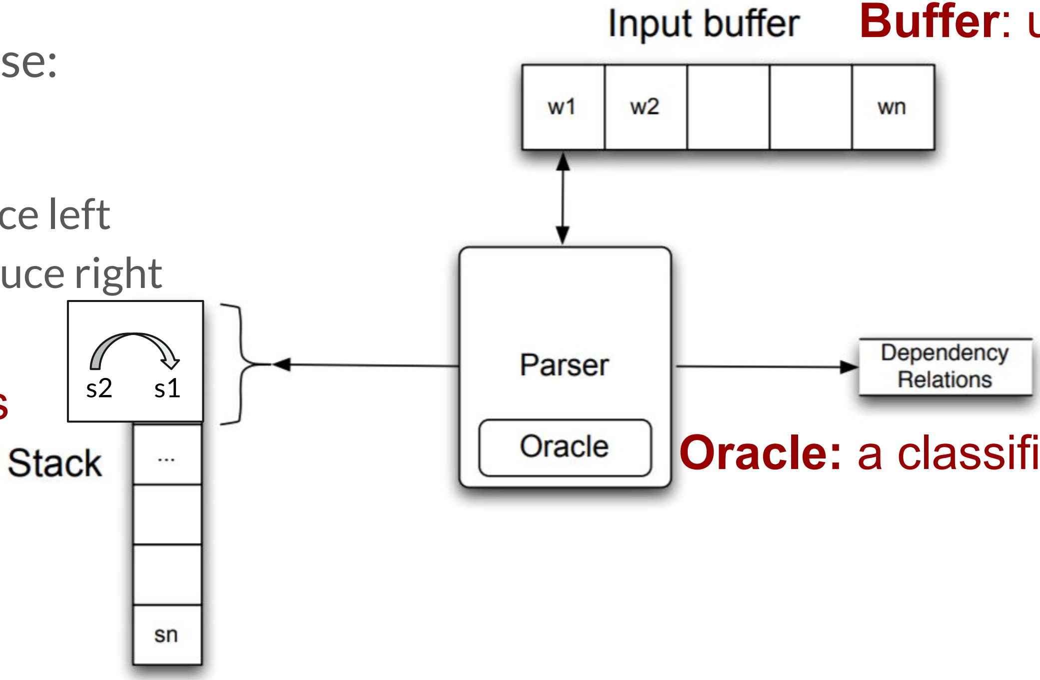
At each step choose:

- Shift
- LeftArc or Reduce left
- RightArc or Reduce right

Stack: partially processed words

Buffer: unprocessed words

Oracle: a classifier



$$C_{\text{accept}} = ([\text{ROOT}], \varnothing, A)$$

Shift-Reduce Parsing

Configuration:

- Stack, Buffer, Oracle, Set of dependency relations

Operations by a classifier at each step:

- Shift
  - remove w1 from the buffer, push it onto the stack as s1
- LeftArc or Reduce left
  - assert a head-dependent relation between s1 and s2 (s1 → s2)
  - pop s1 from the stack; pop s2 from the stack; then push (s2← s1) onto the stack
- RightArc or Reduce right
  - assert a head-dependent relation between s2 and s1 (s2 → s1)
  - pop s1 from the stack; pop s2 from the stack; then push (s2 → s1) onto the stack

Want to see an example of transition-based parsing in action?

Slides 30-44 of this slide deck by Noah Smith do a really nice job of walking through the full transition-based assembly of a sentence's parse visually.