

premetto: c'è sicura la media $NL \approx CF$

Tecnologie del Linguaggio Naturale

Parte Prima

Lezione n. 04-2

Sintassi: la performance (a costituenti)

15-03-2021

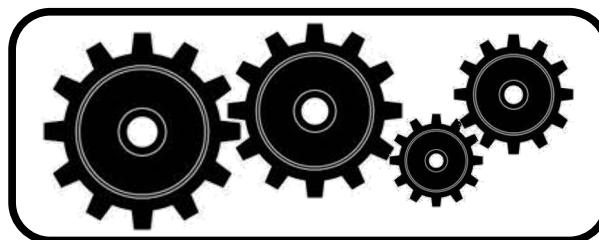
in input: lista di parole
in output: struttura sintattica (norm. un albero) + raggr. le relaz. sintattiche
fra le parole

↑
quali algo. usare + ottenerla?

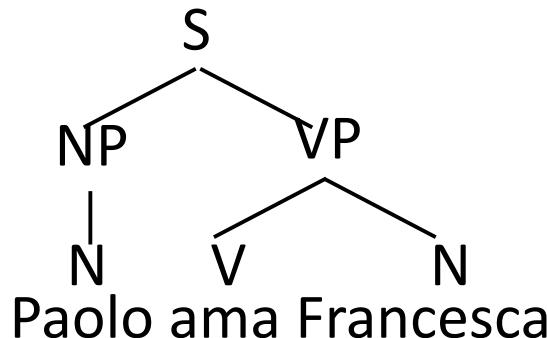
Parser

input

Paolo ama Francesca



output



Outline

- 1 • Parser anatomy
 - 2 • Top-down vs. bottom-up
 - 3 • CKY algorithm: Cocke-Kasami-Younger
 - 4 • Parsing probabilistico e TB grammars
 - 5 • Parsing parziale

Outline

- Parser anatomy
- Top-down vs. bottom-up
- CKY algorithm: Cocke-Kasami-Younger
- Parsing probabilistico e TB grammars
- Parsing parziale

Parser anatomy

Io chiede spesso all'èrone!

il modello di competenza

(1) Grammar (or automa model)

Context-Free, TAG, CCG, Dependency ...

questo è l'ipot. che si sta facendo su la formalità linguistica delle sintassi

(2) Algorithm

I. Search strategy

top-down, bottom-up, left-to-right, ...

come costruisce le strutt. dati

II. Memory organization

back-tracking, dynamic programming, beam search, ...

(3) Oracle

Probabilistic, rule-based, ...

che ha le regole S → NT S → NP quale uno prima? chi decide?

ambiguità!

Outline

- Parser anatomy
- Top-down vs. bottom-up
- CKY algorithm: Cocke-Kasami-Younger
- Parsing probabilistico e TB grammars
- Parsing parziale

Grammar

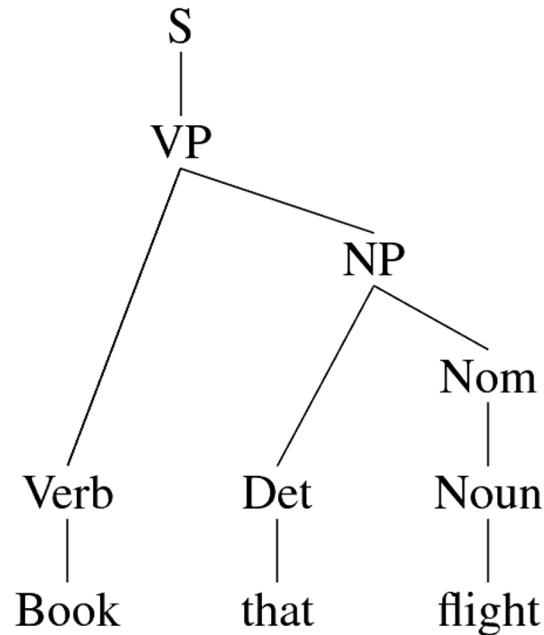
sono fondamentali per questa grammatica definire
anche i PoS taggging. Potrebbe non essere così

1 $S \rightarrow NP VP$
2 $S \rightarrow Aux NP VP$
3 $S \rightarrow VP$
 $NP \rightarrow Det Nominal$
 $Nominal \rightarrow Noun$
 $Nominal \rightarrow Noun Nominal$
 $NP \rightarrow Proper-Noun$
 $VP \rightarrow Verb$
 $VP \rightarrow Verb NP$

$Det \rightarrow that | this | a$
 $Noun \rightarrow book | flight | meal | money$
 $Verb \rightarrow book | include | prefer$
 $Aux \rightarrow does$ gestendo a questo livello il PoS taggging non riesce a eliminare le ambiguità: book è verb o nome?
 $Prep \rightarrow from | to | on$
 $Proper-Noun \rightarrow Houston | TWA$
 $Nominal \rightarrow Nominal PP$



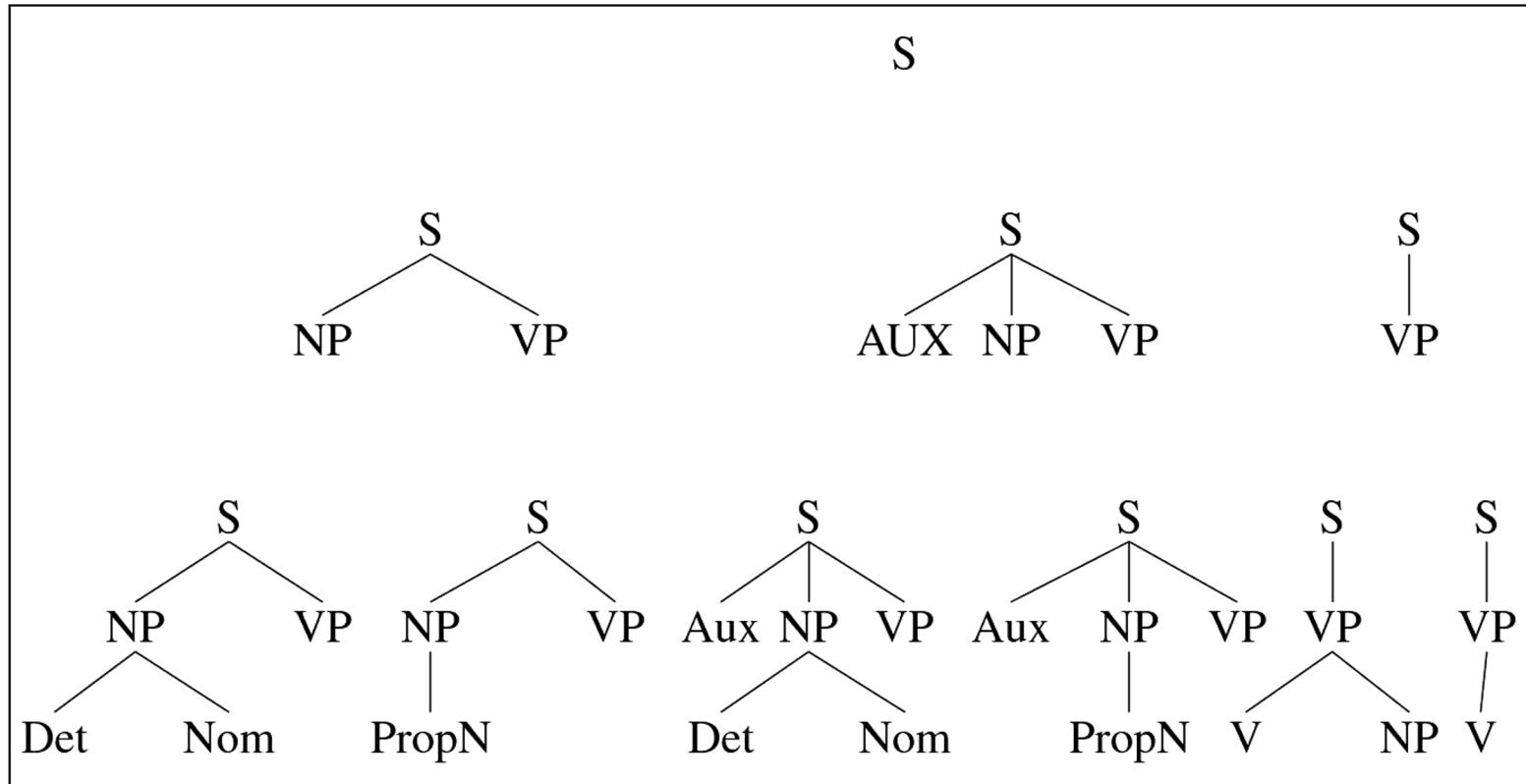
Target Parse



Information sources

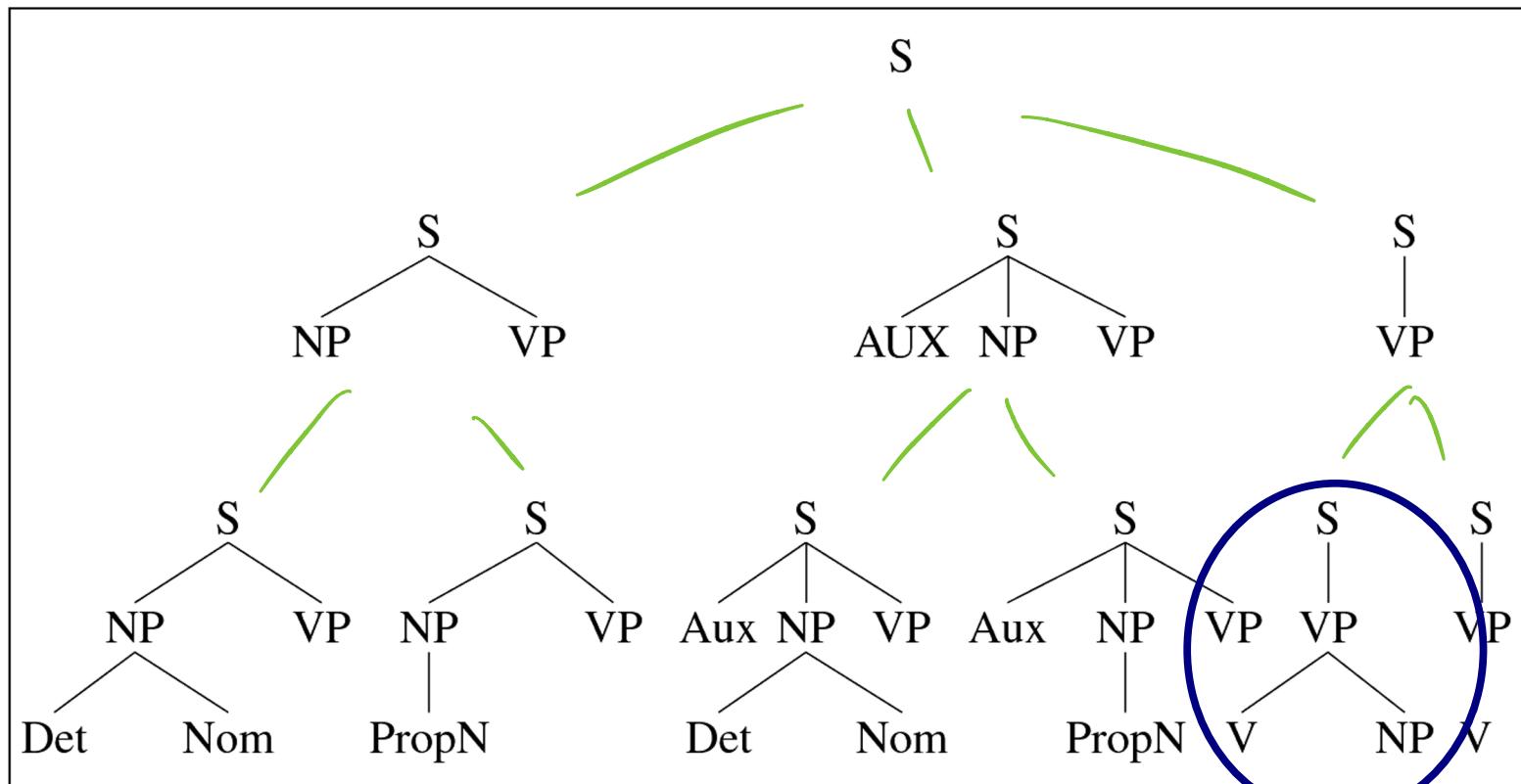
- The grammar -> goal-directed parsing -> top-down
 - Solo ricerche che possono portare a risposte corrette, cioè con radice S
 - ^{WN} Ma comporta la creazione di alberi non compatibili con le parole
 - Razionalisti
- The sentence -> data-directed parsing -> bottom-up
 - Solo ricerche compatabili con le parole
 - ^{WN} Ma comporta la creazione di alberi non corretti
 - Empiricisti

Top-Down



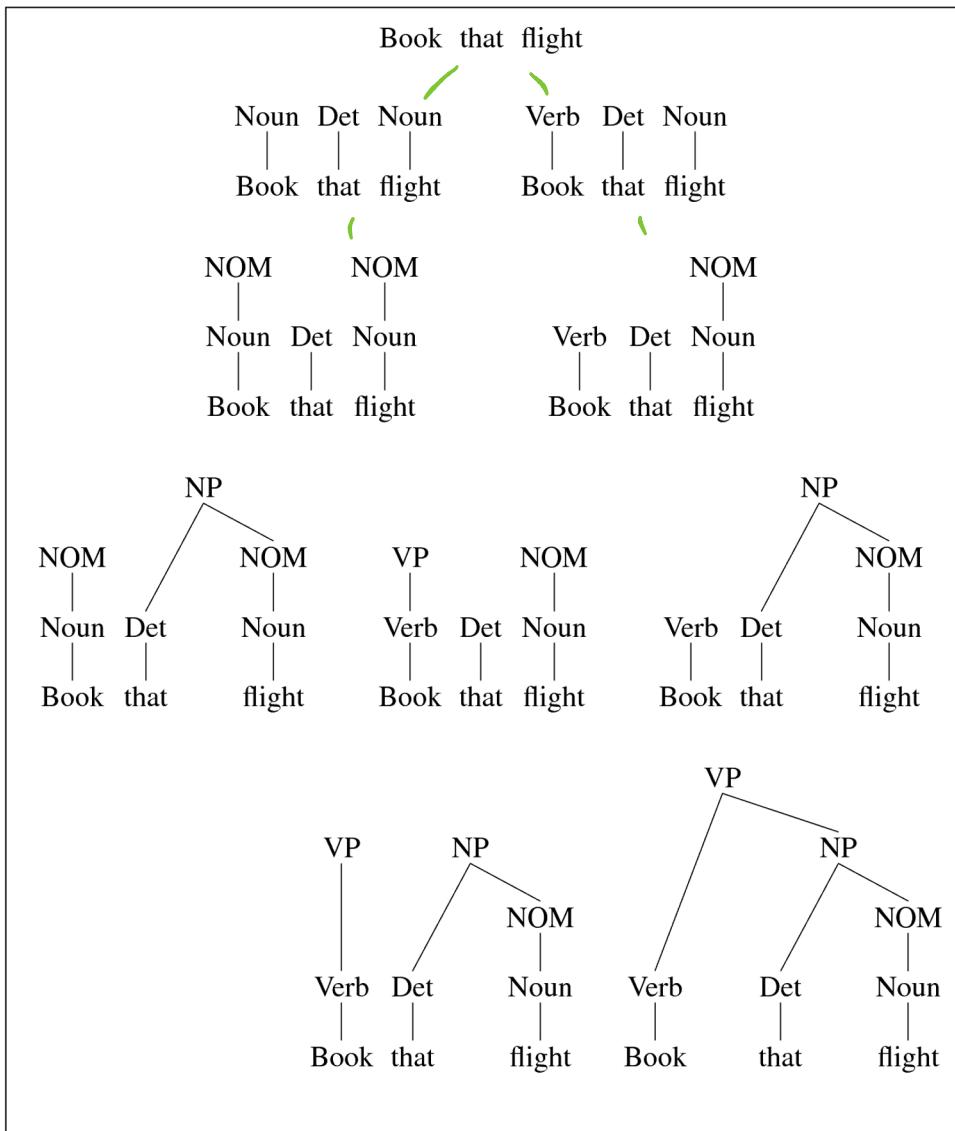
Book that flight

Top-Down

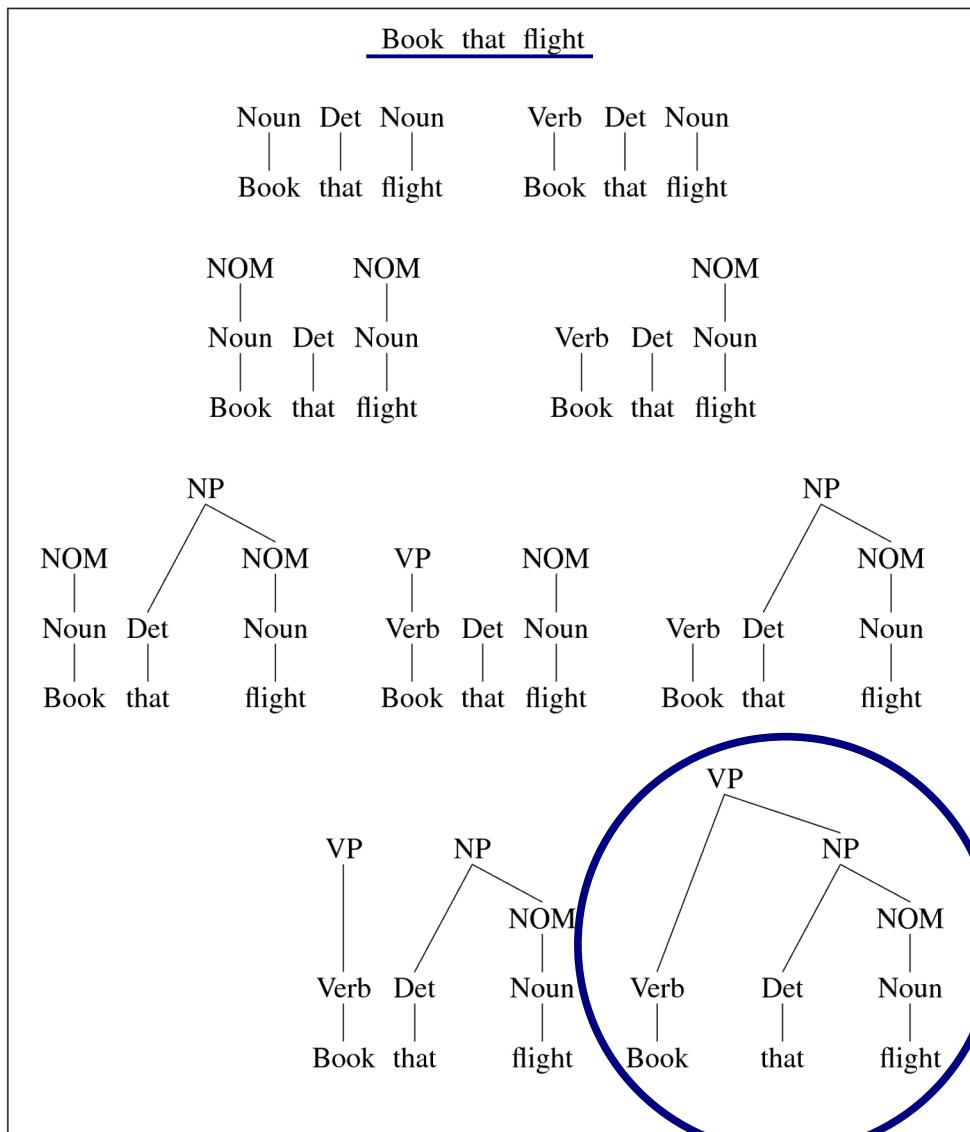


Book that flight

Bottom-Up



Bottom-Up



Parser A

(1) Grammar

Context-Free, ...

(2) Algorithm

I. Search strategy

top-down, bottom-up, left-to-right, ...

II. Memory organization

back-tracking, dynamic programming, ...

(3) Oracle

Probabilistic, rule-based, ...

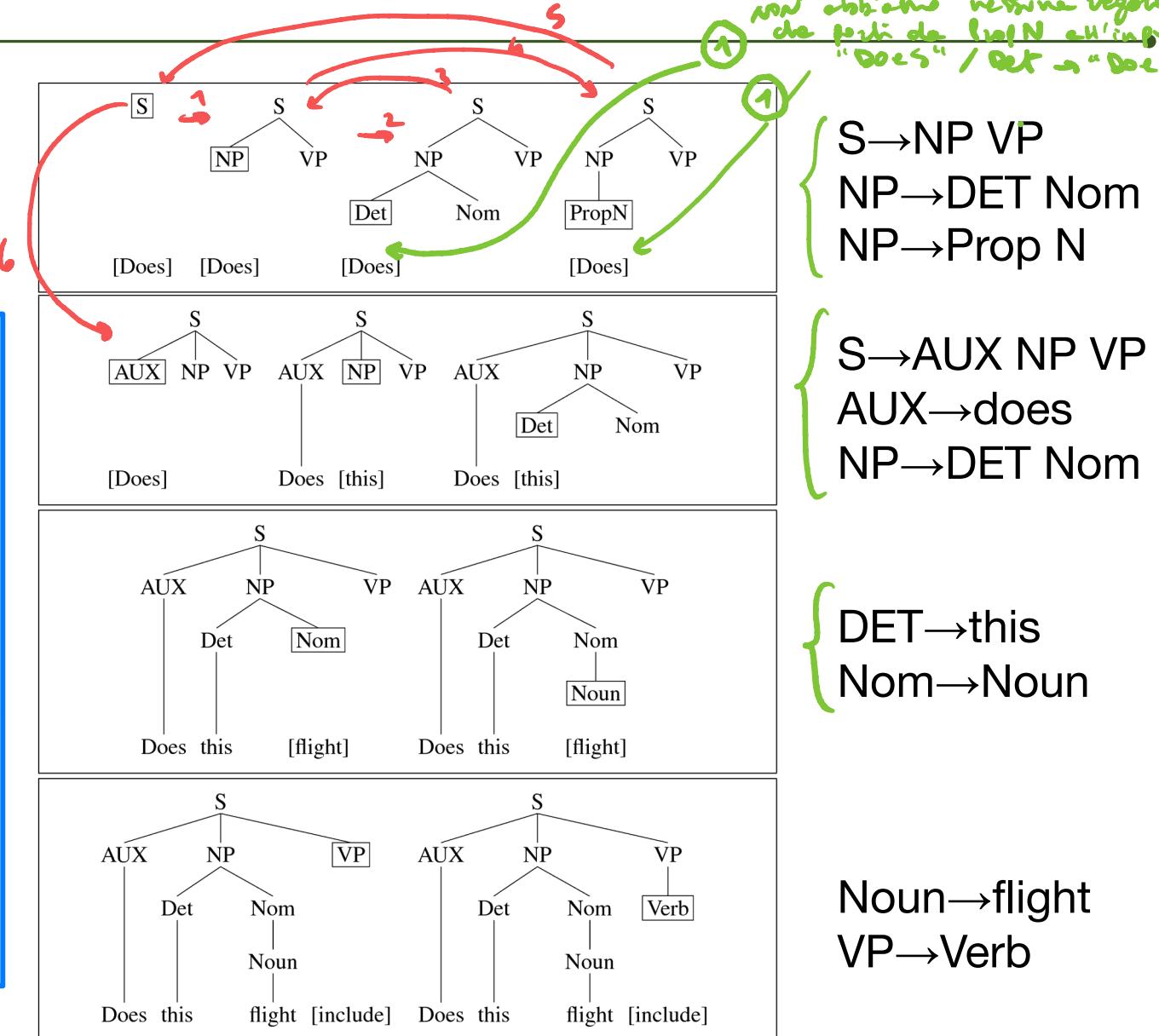
il più chiaro: quando hai + regole, scegli la prima



Parser A - 1

Does this flight include a meal ...

2 Quindi faccio backtracking fino al primo nodo e vi posso applicare nuove regole



$S \rightarrow NP\ VP$
 $S \rightarrow Aux\ NP\ VP$
 $S \rightarrow VP$
 $NP \rightarrow Det\ Nominal$
 $Nominal \rightarrow Noun$
 $Nominal \rightarrow Noun\ Nominal$
 $NP \rightarrow Proper-Noun$
 $VP \rightarrow Verb$
 $VP \rightarrow Verb\ NP$

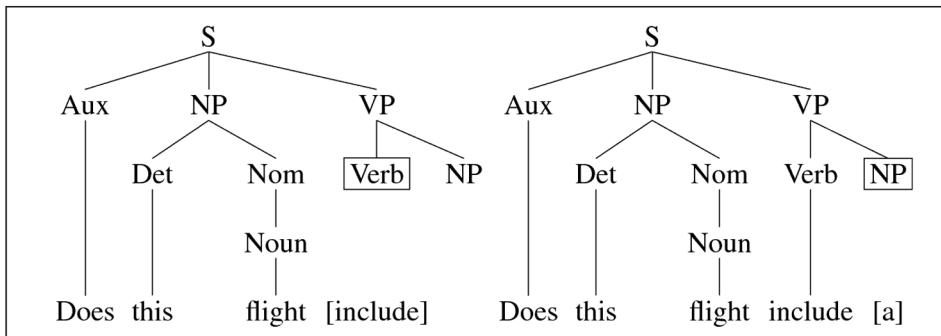
 $Det \rightarrow that\ | \ this\ | \ a$
 $Noun \rightarrow book\ | \ flight\ | \ meal\ | \ money$
 $Verb \rightarrow book\ | \ include\ | \ prefer$
 $Aux \rightarrow does$

 $Prep \rightarrow from\ | \ to\ | \ on$
 $Proper-Noun \rightarrow Houston\ | \ TWA$

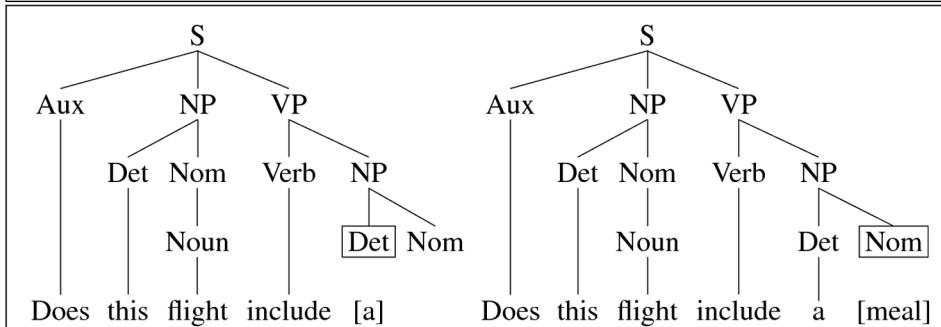
 $Nominal \rightarrow Nominal\ PP$

Parser A - 2

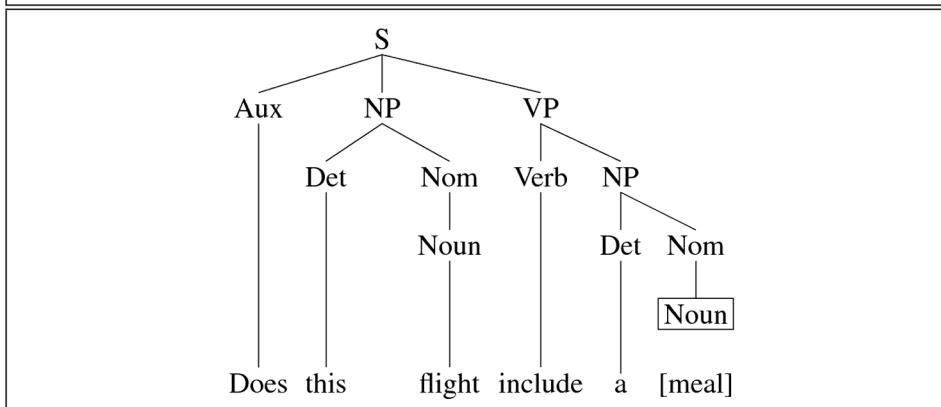
Finire a una
soluzione? Sì, ma:
con 1 left-rewriting
2 shared
sub-problems



$VP \rightarrow \text{Verb NP}$
 $\text{Verb} \rightarrow \text{include}$

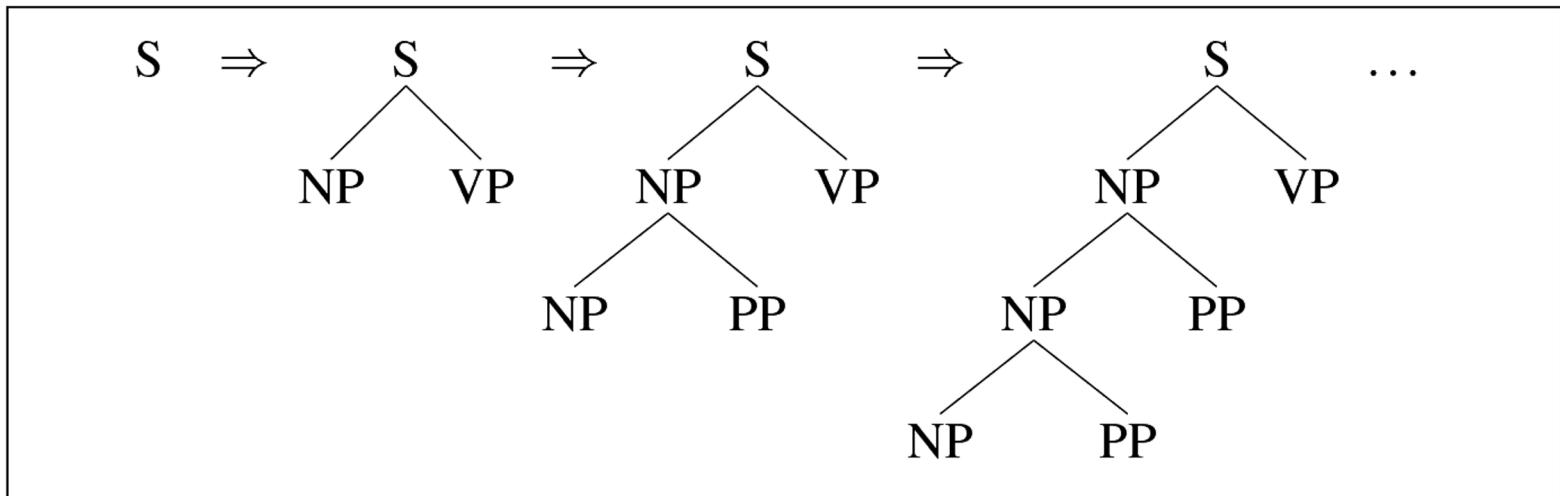


$NP \rightarrow \text{Det Nom}$
 $\text{Det} \rightarrow a$



$\text{Nom} \rightarrow \text{Noun}$
 $\text{Noun} \rightarrow \text{meal}$

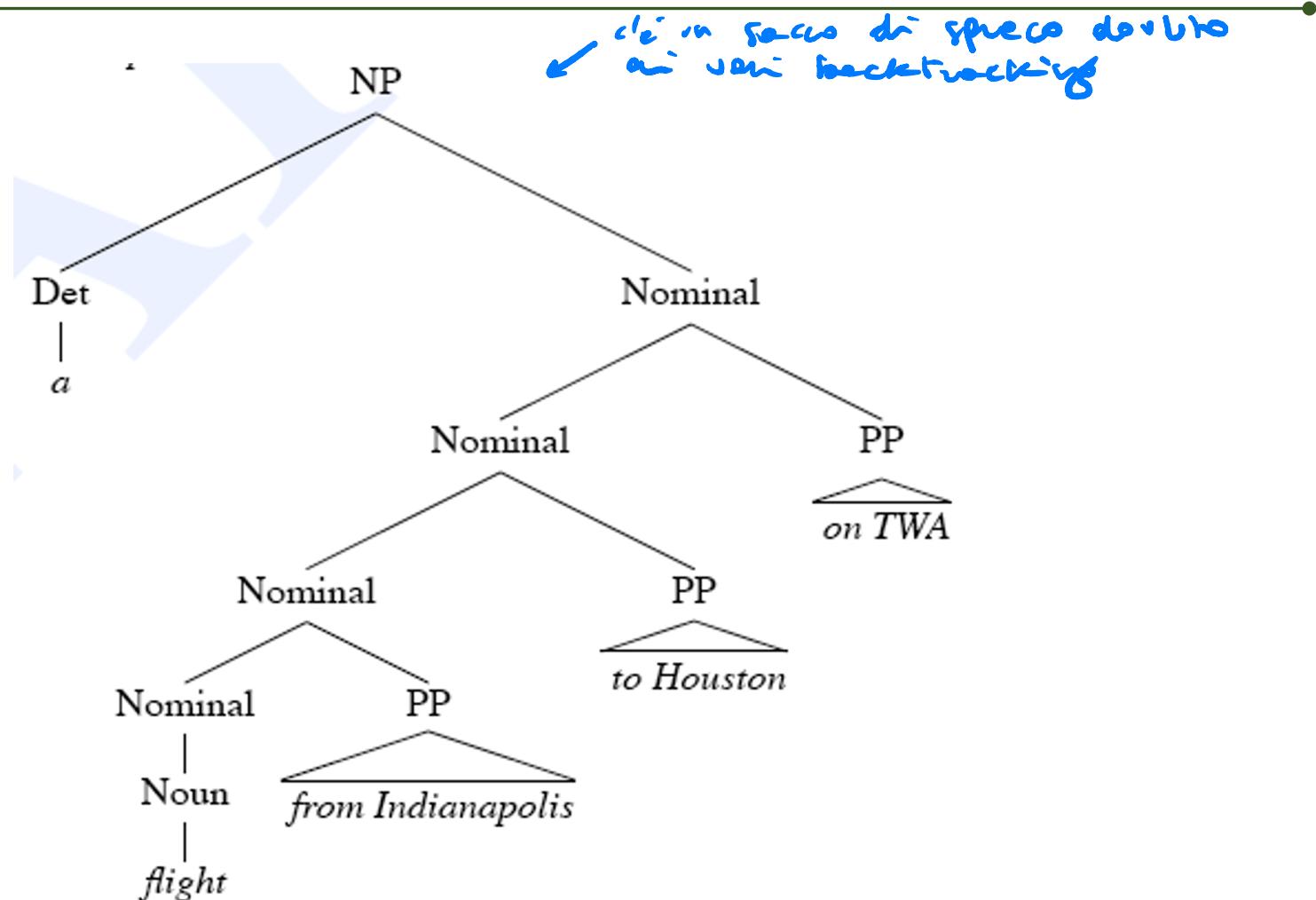
① Left-Recursion

$$\overbrace{NP \rightarrow NP\ PP}^{\downarrow}$$


↑
se in loop

⑦

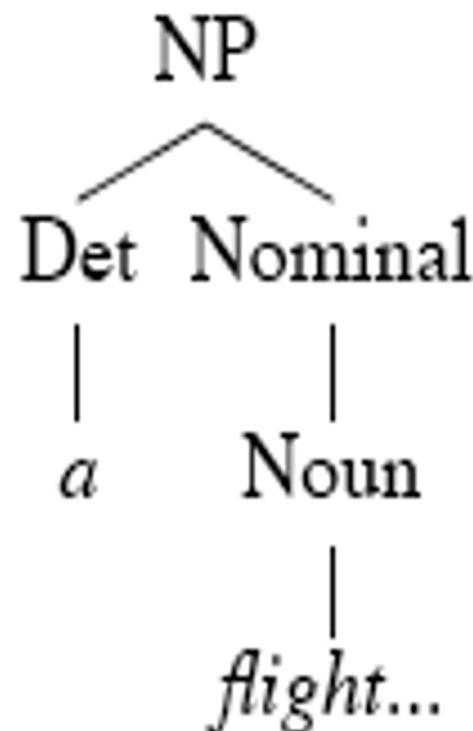
Subtree parsing -> Shared Sub-problems



Shared Sub-problems

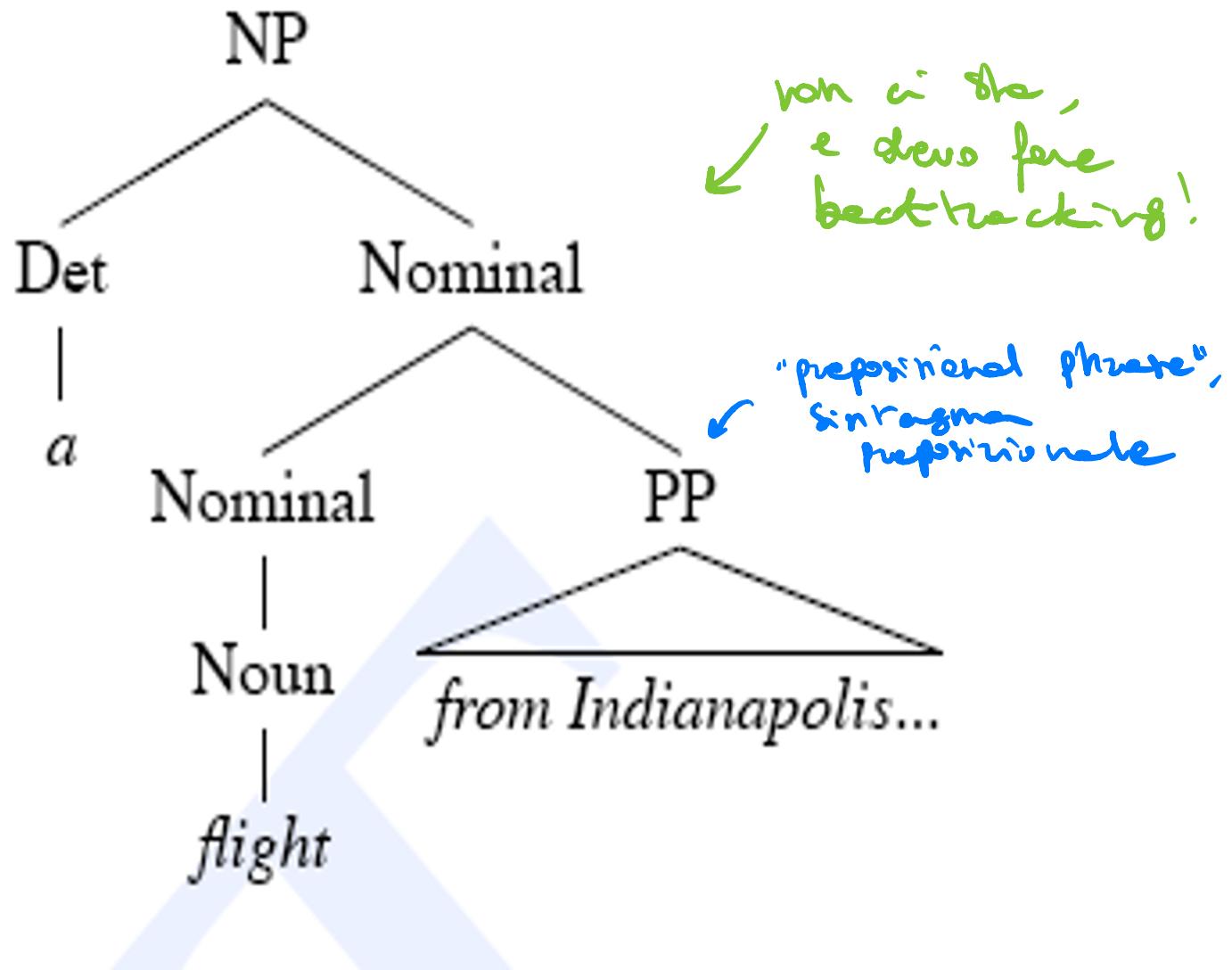
- Top-down parse che ha già espanso la NP
- 2 possibili continuazioni:
 - Nominal -> Noun
 - Nominal -> Nominal PP
- Se scelgo sempre la prima ottengo ...

Shared Sub-problems

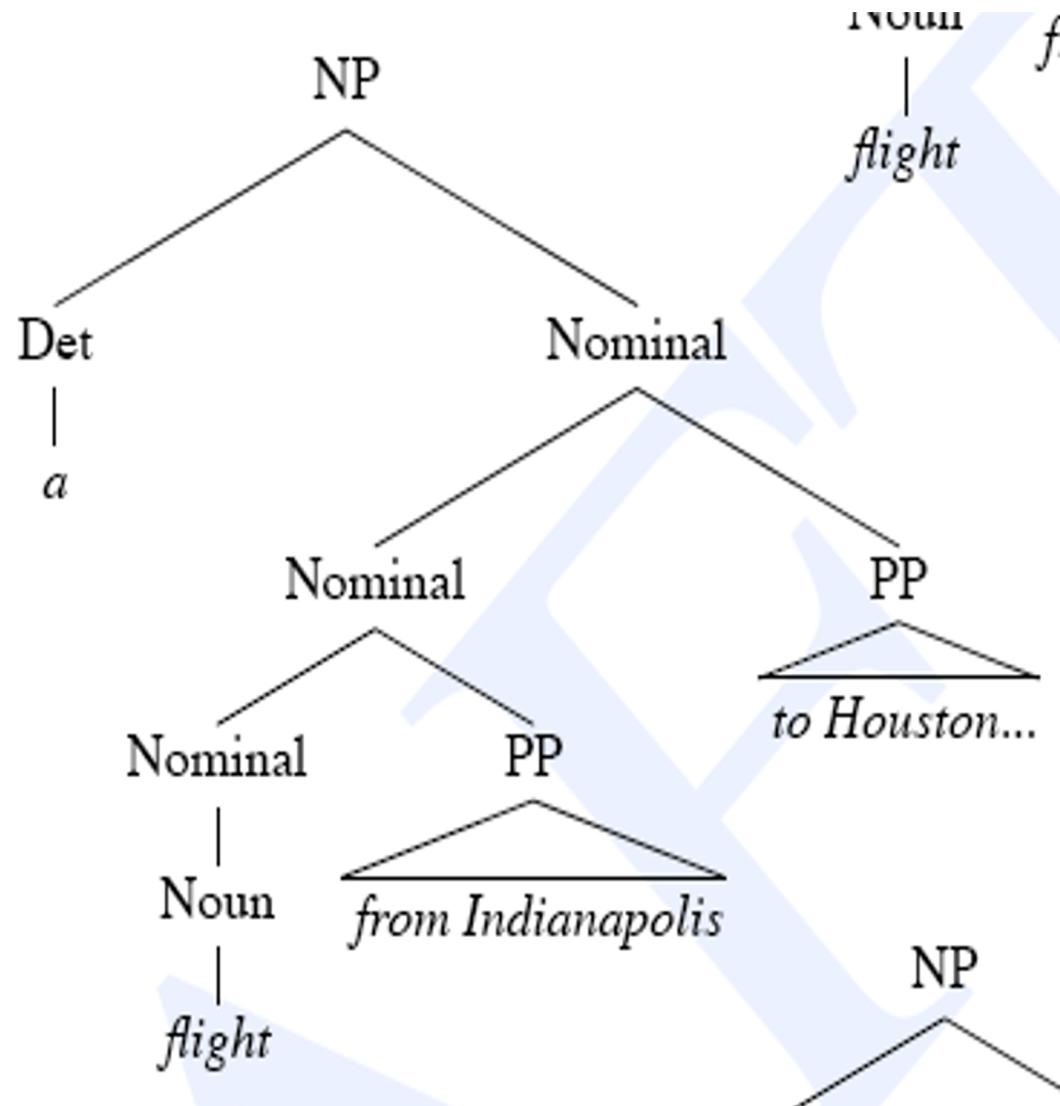


↓
poi compare "from Indianapolis"
(e left-to-right) ↓

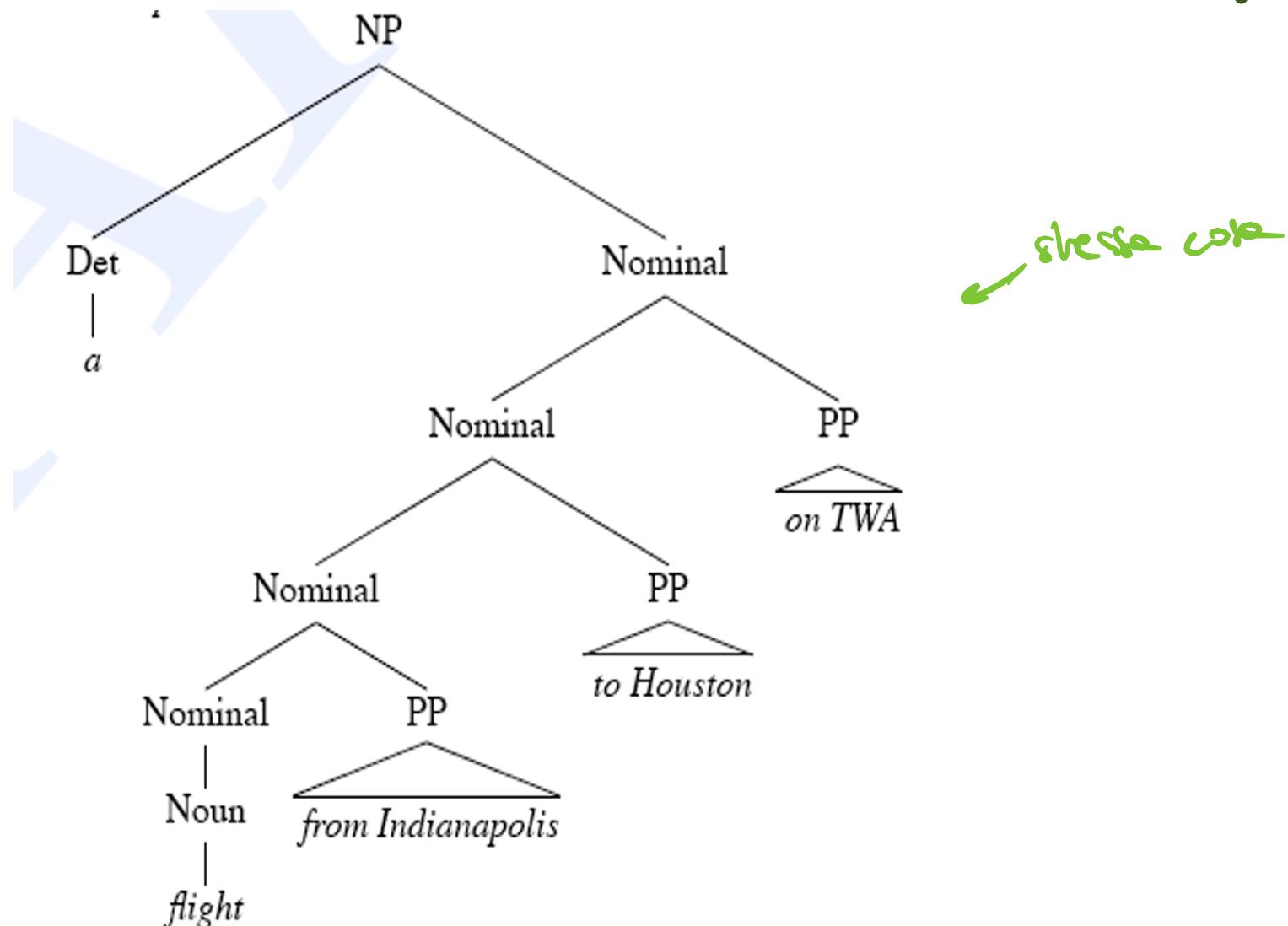
Shared Sub-problems



Shared Sub-problems



Shared Sub-problems



Structural ambiguity

- One sentence can have several “legal parse tree”
sia parziale che completa
 - Structural (cf. PoS ambiguity)
 - attachment ambiguity
 - coordination ambiguity
 - 15 words $\Rightarrow \sim 1000000$ parse trees
- le grammatiche hanno
2 fonti di ambiguità
sintattica*

① Attachment ambiguity (PP)

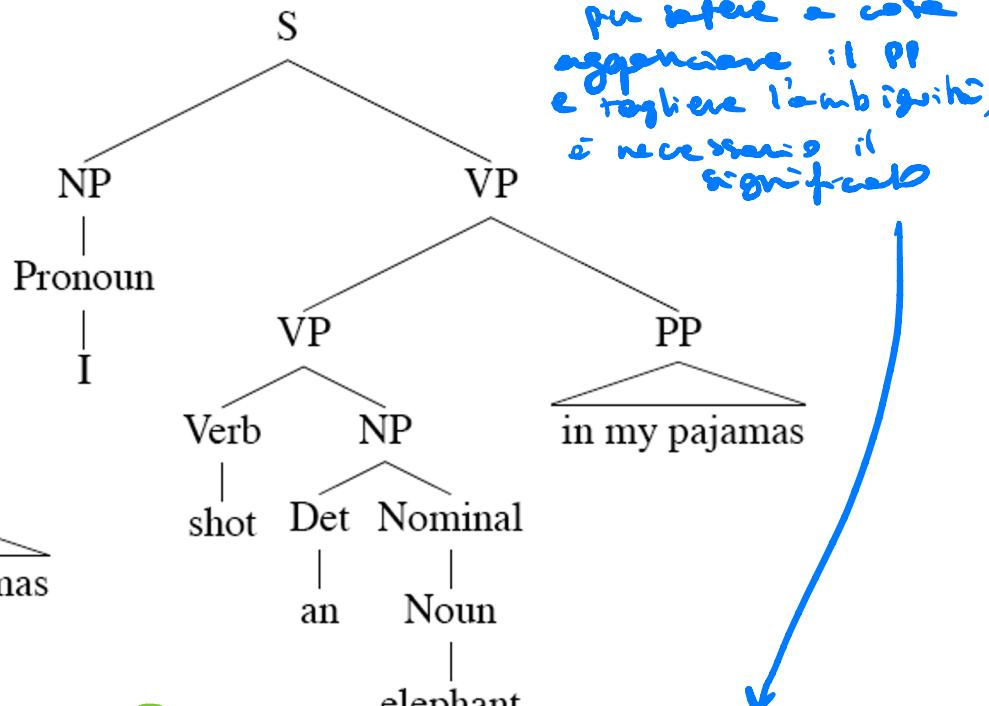
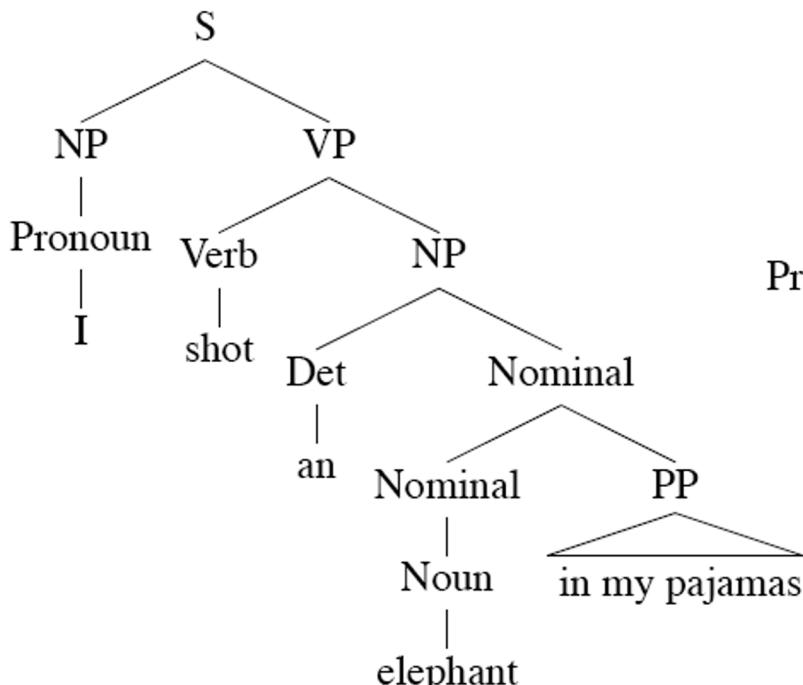
One morning I shot an elephant in my pajamas.

How he got into my pajamas I don't know.

Groucho Marx, *Animal Crackers*, 1930

L'ambiguità sintattica può essere risolta solo sul livello semantico

?



"ho mangiato la pizza con le acciughe" vs "ho mangiato la pizza con le fette"

Il livello sintattico e lessicale possono avere ambiguità diverse.

Ci sono frasi ambigue a livello lessicale e non a livello sintattico

"tutti gli uomini amano una donna"

② Coordination ambiguity

... televisioni e radio nazionali ...

Si può mangiare e bere qualcosa

Si può mangiare e pagare qualcosa

↑
il linguaggio umano è compatto, non
c'è una verbosità grande

Coordination ambiguity

Why punctuation matters.

Some people find inspiration in cooking their families and their dogs.

Others find inspiration in cooking, their families, and their dogs.



som~~e~~ecards
user card

www.writerswrite.co.za

Intervallo: Humor and ambiguity

Tichiti Song

- Ho vinto un soggiorno a Parigi e non ho il camion per andarlo a prendereeee

ma anche:

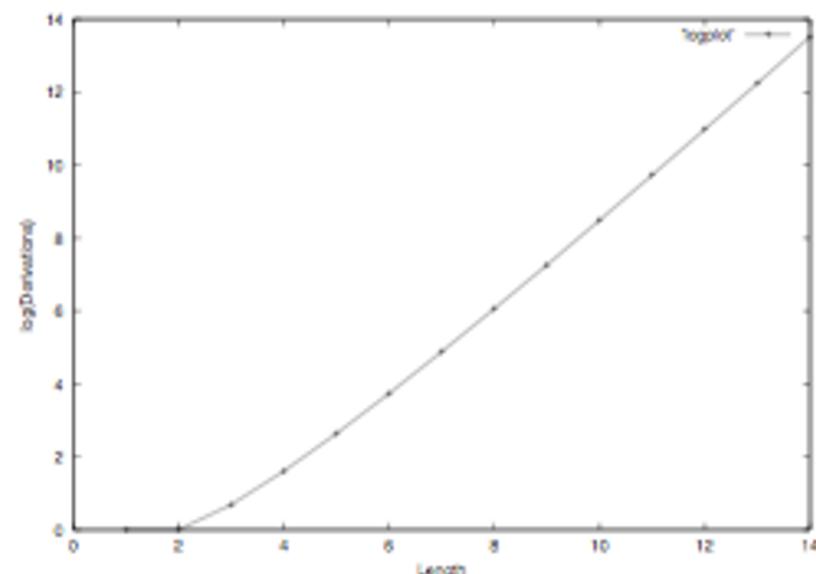
- La donna è mobile, ed io mi sento mobiliere.
- Una ragazza mi ha strizzato l'occhio: mi han dato tre punti
- C'era un signore per strada, mi ha chiesto qual era il marciapiede di
fronte, gli ho detto: "E' quello la"" e lui mi ha risposto: "Ero la' e mi hanno
mandato qua!"

Structural ambiguity

- CFG rules { $S \rightarrow S S$, $S \rightarrow a$ }

↳ grammar is super ambiguous

#a	# parses
1	1
2	1
3	2
4	5
5	14
6	42
7	132
8	429
9	1430
10	4862
11	16796



~ esplosione combinatoria

Structural ambiguity

veloce ..

Catalan numbers (Church and Patil 1982)

- Serie potenza

- $N \rightarrow \text{natural} \mid \text{language} \mid \text{processing} \mid \text{course}$
- $N \rightarrow NN$

$$C_n = \frac{1}{n+1} \binom{2n}{n} = \frac{(2n)!}{(n+1)! n!} = \prod_{k=2}^n \frac{n+k}{k} \quad \text{for } n \geq 0.$$

- Intuizione

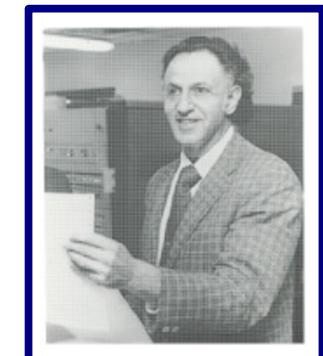
- ugual numero di parentesi aperte e chiuse
- propriamente incapsulate, i.e. un'aperta precede una chiusa

Come gestire l'esplosione combinatoria?



Dynamic Programming

- **Richard Bellman, 1957** Principio di ottimalità: “An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision”.
 - Sottostrutture ottimali *(una volta che sono: prefissi qui sono "": suffisso)*
 - Sottoproblemi sovrapponibili
 - Memoizzazione
- Simile a dividi et impera



Come gestire l'esplosione combinatoria?

Dynamic Programming Parsing

- CKY algorithm: calcola tutti i possibili parse in tempo $O(n^3)$. Il difetto maggiore è che il caso peggiore e il caso medio coincidono (ma anche esige una forma normale per la grammatica).
- Earley algorithm: calcola tutti i possibili parse in tempo $O(n^3)$ per una CFG arbitraria, $O(n^2)$ per una CFG ambigua, e $O(n)$ per una bounded-state CFG (e.g. $S \rightarrow aSa \mid bSb \mid aa \mid bb$). Il caso medio ha una performance migliore rispetto a CKY.
- Deterministic parsing: solo un parsing -> ambiguità limitata -> programming language parsing
 - top-down: e.g. LL parsing
 - bottom-up: LR or shift-reduce parsing

Outline

- Parser anatomy
- Top-down vs. bottom-up
- CKY algorithm: Cocke-Kasami-Younger
- Parsing probabilistico e TB grammars
- Parsing parziale

Parser B (CKY- Cocke Kasamy Younger)

R

(1) Grammar

Context-Free, ...

non c'è left recursion
non ci sono diverse riferenze che portano
a una stessa
suspension exponentiale

(2) Algorithm

I. Search strategy

top-down, bottom-up, left-to-right, ...

II. Memory organization

back-tracking, dynamic programming, ...

use one matrix

(3) Oracle

Probabilistic, rule-based, ...

regole non lexicale

CKY idea

premesse: le grammatiche qui sono false:

- $A \rightarrow BC$: 1 NT sulla sx e 2 NT sulla dx

oppure

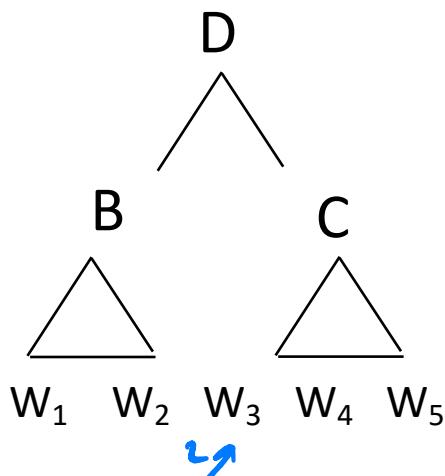
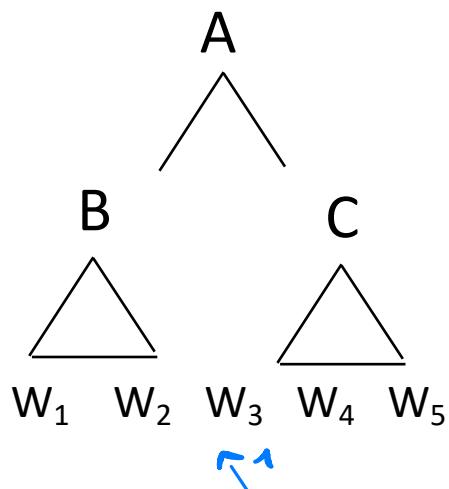
- $D \rightarrow \text{flight}$: 1 simb. NT sulla sx e 1 simb. terminale sulla dx

regole lexicale

! non perdo in generalità: qualsiasi grammatica CF
può essere trasformata in una CF forma norm. di Chomsky
equivalente

$$A \rightarrow BC$$

$$D \rightarrow BC$$



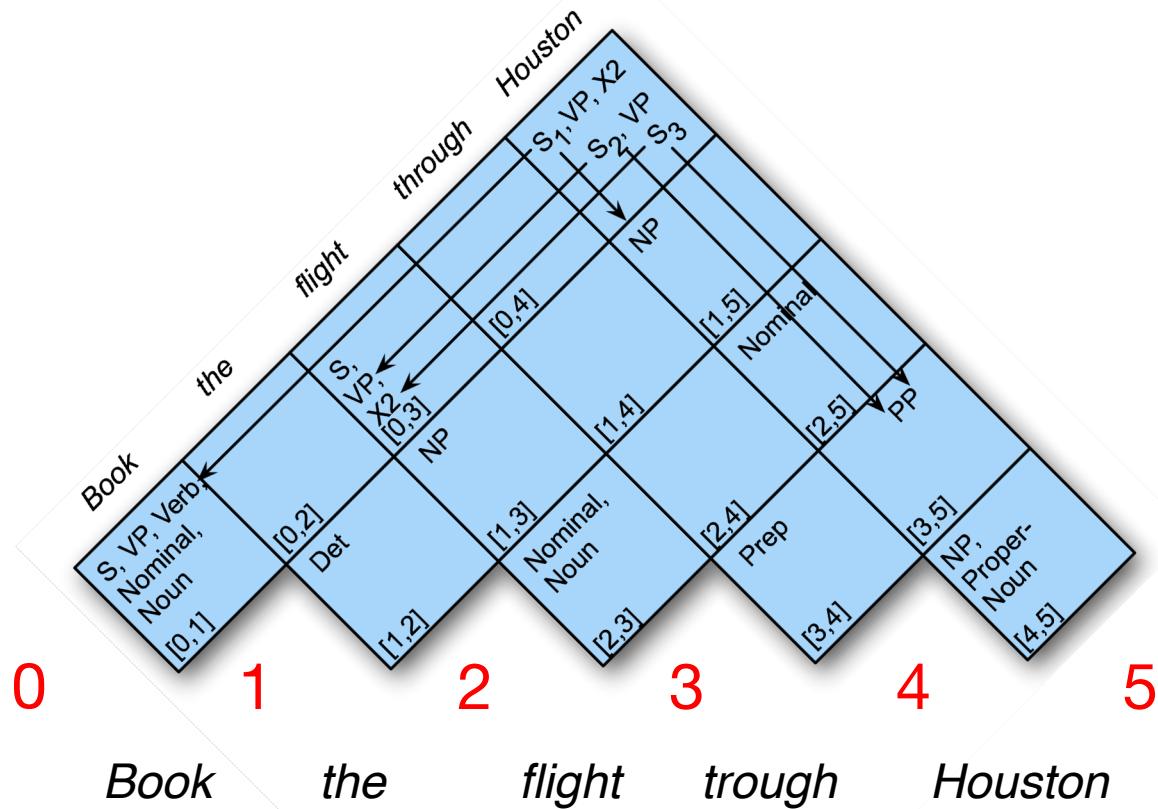
con questo
algoritmo

CF in Forma Normale di Chomsky

- Copy all conforming rules to the new grammar unchanged,
- Convert terminals within rules to dummy non-terminals,
- Convert unit-productions,
- Binarize all rules and add to new grammar.
- Epsilon free

! se mi accorgo che non è la frase qua, rimuovo che per backtracking e
butto tutto, mantiengo i soli alberi come B C

Idea della table

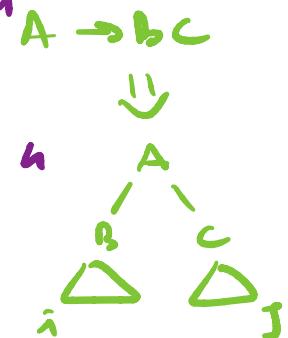


CKY

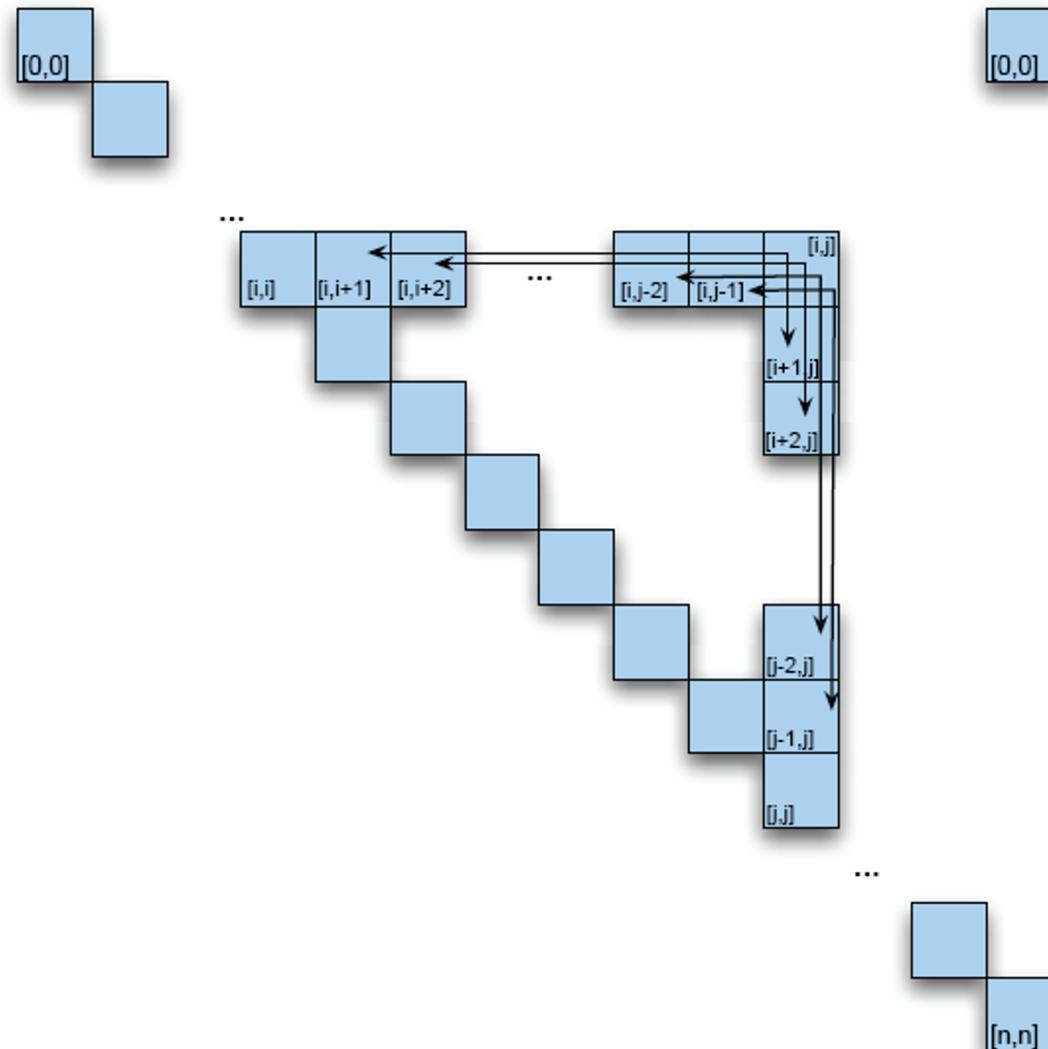
$0Book_1that_2flight_3$

A $\rightarrow BC$

- Se c'è un A allora c'è un B seguito da un C
- Se A va da i to j, c'è un k tale che $i < k < j$
- Cioè B va da i to k e C va da k to j
- Usiamo una table (array bidimensionale) per mettere B in table[i,k], C in table[k,j], A in table[i,j]
- Loop su k
- Se il parsing ha successo $\rightarrow S$ in [0,n]



Filling the table



Grammatica L1

Grammar	Lexicon
$S \rightarrow NP\ VP$	$Det \rightarrow that this a$
$S \rightarrow Aux\ NP\ VP$	$Noun \rightarrow book flight meal money$
$S \rightarrow VP$	$Verb \rightarrow book include prefer$
$NP \rightarrow Pronoun$	$Pronoun \rightarrow I she me$
$NP \rightarrow Proper-Noun$	$Proper-Noun \rightarrow Houston NWA$
$NP \rightarrow Det\ Nominal$	$Aux \rightarrow does$
$Nominal \rightarrow Noun$	$Preposition \rightarrow from to on near through$
$Nominal \rightarrow Nominal\ Noun$	
$Nominal \rightarrow Nominal\ PP$	
$VP \rightarrow Verb$	
$VP \rightarrow Verb\ NP$	
$VP \rightarrow Verb\ NP\ PP$	
$VP \rightarrow Verb\ PP$	
$VP \rightarrow VP\ PP$	
$PP \rightarrow Preposition\ NP$	

Grammatica L1 e normal form

$S \rightarrow NP VP$
 $S \rightarrow Aux NP VP$
 $S \rightarrow VP$

$NP \rightarrow Pronoun$
 $NP \rightarrow Proper-Noun$
 $NP \rightarrow Det Nominal$
 $Nominal \rightarrow Noun$
 $Nominal \rightarrow Nominal Noun$
 $Nominal \rightarrow Nominal PP$
 $VP \rightarrow Verb$
 $VP \rightarrow Verb NP$
 $VP \rightarrow Verb NP PP$
 $VP \rightarrow Verb PP$
 $VP \rightarrow VP PP$
 $PP \rightarrow Preposition NP$

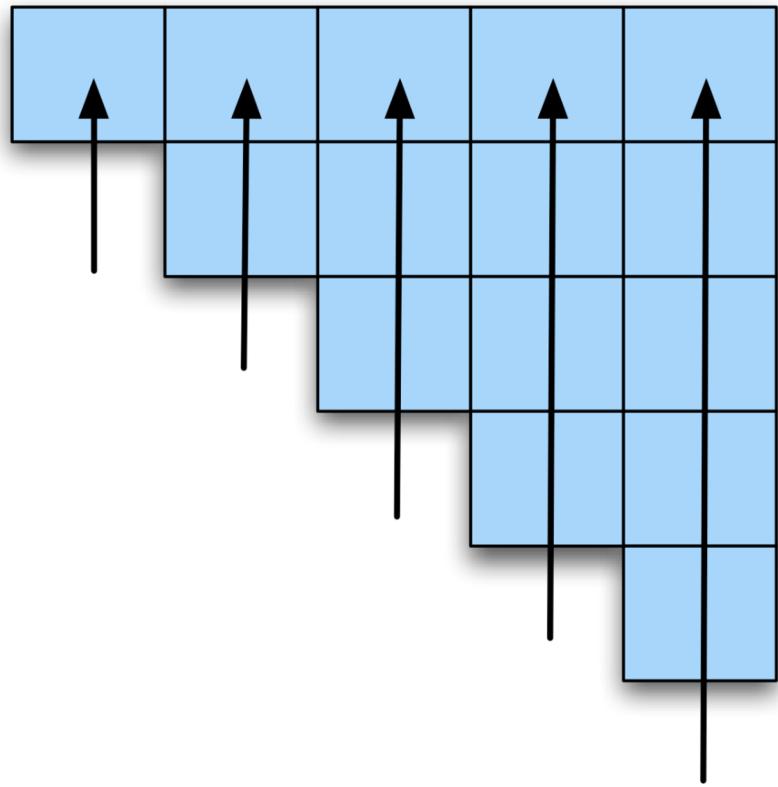
$S \rightarrow NP VP$
 $S \rightarrow XI VP$
 $XI \rightarrow Aux NP$
 $S \rightarrow book | include | prefer$
 $S \rightarrow Verb NP$
 $S \rightarrow X2 PP$
 $S \rightarrow Verb PP$
 $S \rightarrow VP PP$
 $NP \rightarrow I | she | me$
 $NP \rightarrow TWA | Houston$
 $NP \rightarrow Det Nominal$
 $Nominal \rightarrow book | flight | meal | money$
 $Nominal \rightarrow Nominal Noun$
 $Nominal \rightarrow Nominal PP$
 $VP \rightarrow book | include | prefer$
 $VP \rightarrow Verb NP$
 $VP \rightarrow X2 PP$
 $X2 \rightarrow Verb NP$
 $VP \rightarrow Verb PP$
 $VP \rightarrow VP PP$
 $PP \rightarrow Preposition NP$

↑
grammatica cf

↑ forma normale

Book the flight through Houston

S, VP, Verb Nominal, Noun [0,1]	[0,2]	S,VP,X2 [0,3]	[0,4]	S,VP,X2 [0,5]
Det [1,2]	NP [1,3]	[1,4]	NP [1,5]	
	Nominal, Noun [2,3]	[2,4]	Nominal [2,5]	
	Prep [3,4]	PP [3,5]		
		NP, Proper- Noun [4,5]		



CKY Algorithm

```
function CKY-PARSE(words, grammar) returns table
    initializ. for  $j \leftarrow 1$  to LENGTH(words) do
        diagonale (cpz. variabili)  $\rightarrow$   $table[j - 1, j] \leftarrow \{A \mid A \rightarrow words[j] \in grammar\}$ 
        for  $i \leftarrow$  from  $j - 2$  downto 0 do
            for  $k \leftarrow i + 1$  to  $j - 1$  do
                 $table[i, j] \leftarrow table[i, j] \cup$ 
                 $\{A \mid A \rightarrow BC \in grammar,$ 
                     $B \in table[i, k],$ 
                     $C \in table[k, j]\}$ 
```

CKY Algorithm

function CKY-PARSE(*words*, *grammar*) **returns** *table*

for *j* **← from** 1 **to** LENGTH(*words*) **do**

Looping over the columns

table[*j* - 1, *j*] **←** {*A* | *A* \rightarrow *words*[*j*] \in

Filling the bottom cell

for *i* **← from** *j* - 2 **downto** 0 **do**

Filling row *i* in column *j*

for *k* **←** *i* + 1 **to** *j* - 1 **do**

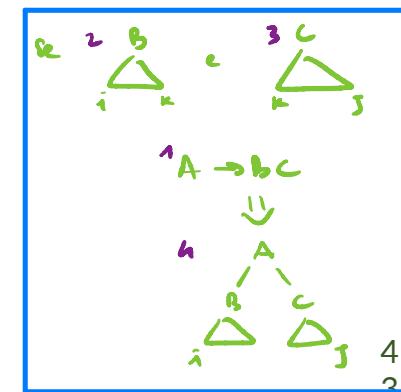
Looping over the possible split
locations between *i* and *j*.

table[*i*, *j*] **←** *table*[*i*, *j*] \cup

{*A* | *A* \rightarrow *BC* \in grammar,

B \in *table*[*i*, *k*],

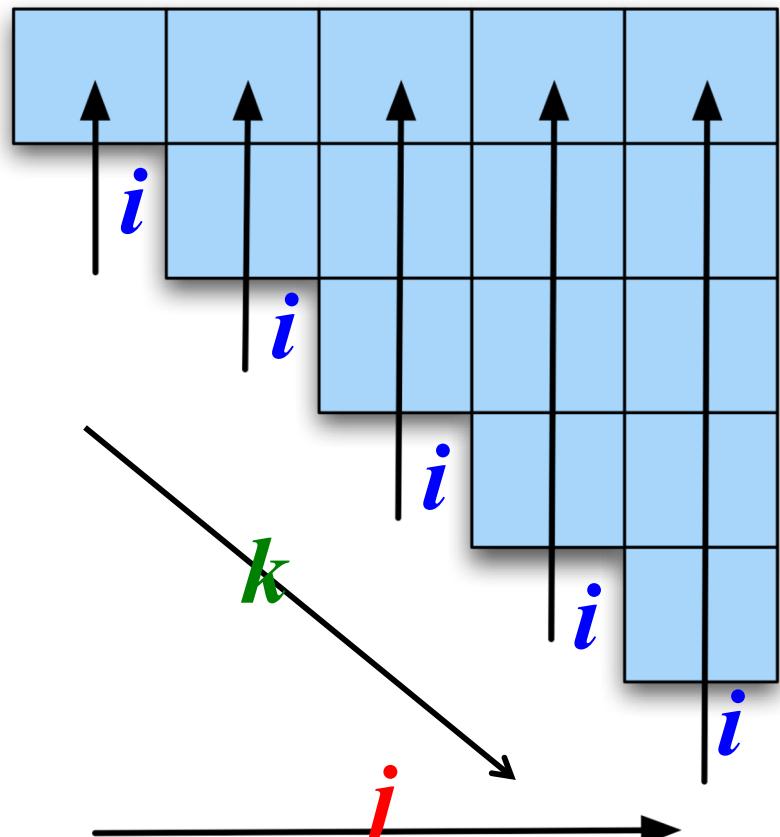
C \in *table*[*k*, *j*]}
Check the grammar for rules
that link the constituents in
[*i*, *k*] with those in [*k*, *j*]. For
each rule found store the
LHS of the rule in cell [*i*, *j*].



$$[j : 1..n \ [i : j-2..0 \ [k : i+1..j-1]]]$$

Book the flight through Houston

0_0	S, VP, Verb Nominal, Noun [0,1]		S,VP,X2 [0,3]		S,VP,X2 [0,5]
1	Det [1,2]	NP [1,3]			NP [1,5]
2		Nominal, Noun [2,3]			Nominal [2,5]
3			Prep [3,4]	PP [3,5]	
4				NP, Proper- Noun [4,5]	
1 2 3 4 5				5	



• $[j : 1 \dots n] \left[i : j-2 \dots 0 \left[k : i+1 \dots j-1 \right] \right]$

Book the flight through Houston

S, VP, Verb, Nominal, Noun [0,1]		S,VP,X2 [0,3]		
	Det [1,2]	NP [1,3]		NP [1,5]
		Nominal, Noun [2,3]		
			Prep [2,4]	PP 1 4 [2,5]
				PP → Preposition NP
			2 [3,4]	3 NP, Proper- Noun [4,5]

$j=5$

$i=3$

$k=4$

Consideriamo solo la colonna 5

O $[j : 1 \dots n \ [i : j-2 \dots 0 \ [k : i+1 \dots j-1]]]$

Book the flight through Houston

S, VP, Verb, Nominal, Noun [0,1]		S,VP,X2 [0,3]		
Det [1,2]	NP [1,3]			NP [1,5]
	Nominal, Noun [2,3]	←		Nominal ↓ [2,5]
		[2,4]		
	Prep [3,4]		PP [3,5]	
				NP, Proper- Noun [4,5]

j=5

i=2

k=3

$$\Theta [j : 1 \dots n [i : j-2 \dots 0 [k : i+1 \dots j-1]]]$$

Book the flight through Houston

S, VP, Verb, Nominal, Noun [0,1]		S,VP,X2 [0,3]		
[0,2]		[0,4]		[0,5]
Det ← NP				NP
[1,2]	[1,3]	[1,4]		[1,5]
	Nominal, Noun [2,3]			Nominal [2,5]
		[2,4]		
			Prep [3,4]	PP [3,5]
				NP, Proper- Noun [4,5]

j=5

i=1

k=2

$[j : 1 \dots n] [i : j-2 \dots 0] [k : i+1 \dots j-1]$]]

Reconoscere quale se le
fusse appartenente o
non alle grammatica

è si se alle
fusse da farlo
in $[0,5]$ un "s"

Book the flight through Houston

S, VP, Verb, Nominal, Noun [0,1]		S, VP, X2 [0,3]		S ₁ , VP, X2 S ₂ , VP S ₃
Det [1,2]	NP [1,3]			NP [1,5]
	Nominal, Noun [2,3]			Nominal [2,5]
			Prep [3,4]	PP [3,5]
				NP, Proper- Noun [4,5]

$j=5$

$i=0$

$k=1,3,3$

L'output non è un solo numero, ma
tutta la matrice

vs

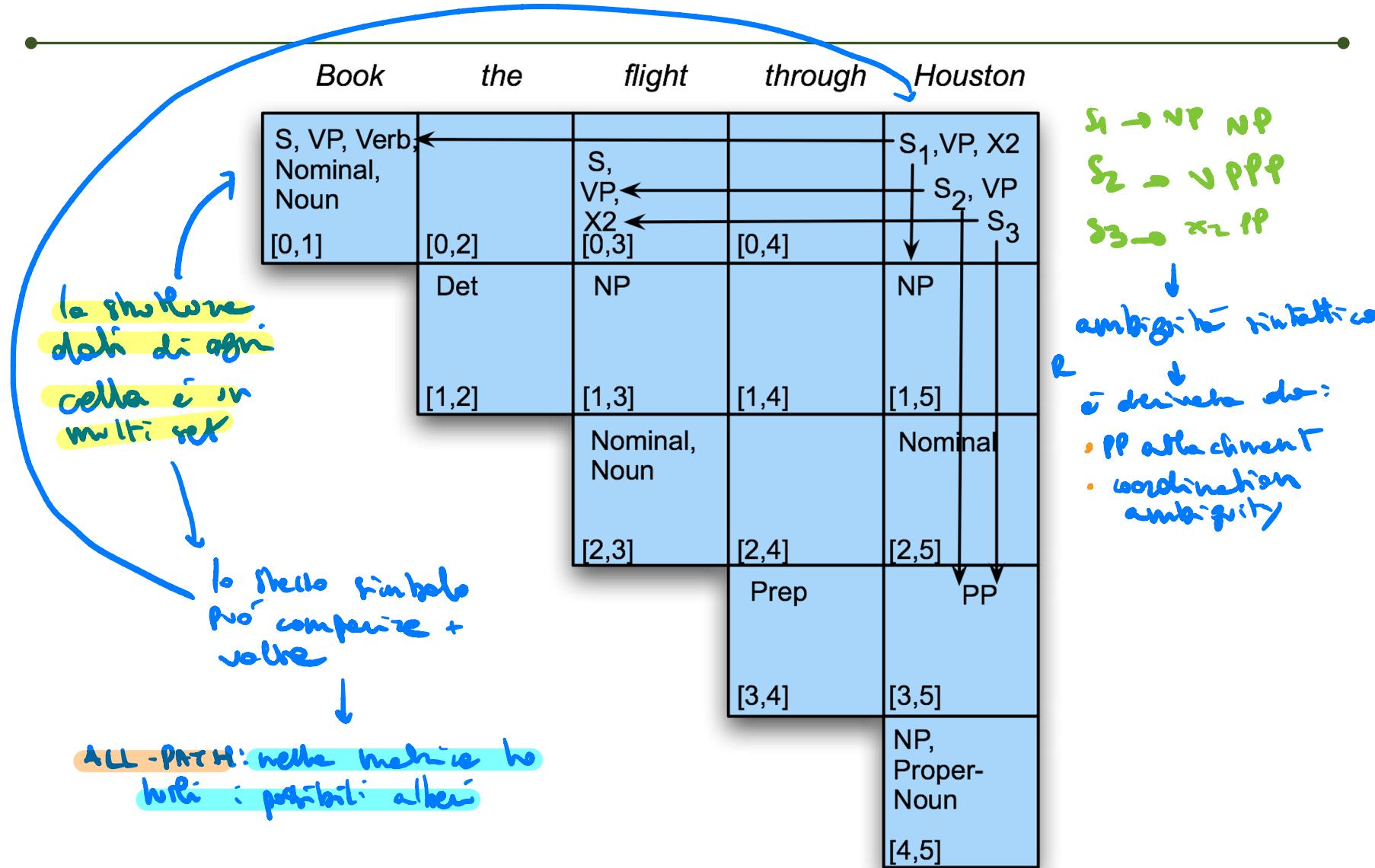
Parseur : fa qualcosa
in più, dice quel c'è
l'albero gusto/gli
alberi-piatti (le passi
potrebbero essere
ambigue)

per questo usa dei
back pointers

Successo!

<i>Book</i>	<i>the</i>	<i>flight</i>	<i>through</i>	<i>Houston</i>
S, VP, Verb Nominal, Noun [0,1]		S,VP,X2 [0,3]	[0,4]	S,VP,X2 [0,5]
	Det [1,2]	NP [1,3]		NP [1,5]
		Nominal, Noun [2,3]		Nominal [2,5]
			Prep [3,4]	PP [3,5]
				NP, Proper- Noun [4,5]

Parsing vs. recognition -> backtrace



Complessità

- $O(n^3)$
- $O(n^5)$ per la versione probabilistica (vedi dopo)
- Troppo lento per il real-time (motori di ricerca)
- Allora: bisogna limitare il # di output
 - pruning probabilistico → beam
 - parziale
 - dipendenze (prossima lezione)

Bisogna provare ...

S → NP VP

NP → Paolo

VP → VP ADV

NP → Francesca

VP → V NP

V → ama

ADV → dolcemente

Paolo₁ama₂Francesca₃dolcemente₄

Outline

- Parser anatomy
- Top-down vs. bottom-up
- CKY algorithm: Cocke-Kasami-Younger
- **Parsing probabilistico e TB grammars**
- Parsing parziale

Parser C (CKY probabilistico)

(1) Grammar

Context-Free, ...

(2) Algorithm

I. Search strategy

top-down, bottom-up, left-to-right, ...

II. Memory organization

back-tracking, dynamic programming
(+beam?), ...

(3) Oracle

Probabilistic, rule-based, ...



CKY probabiistico

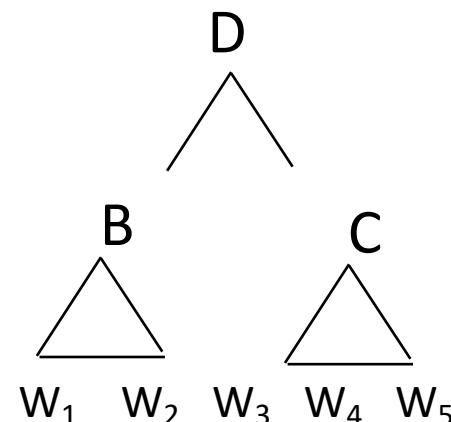
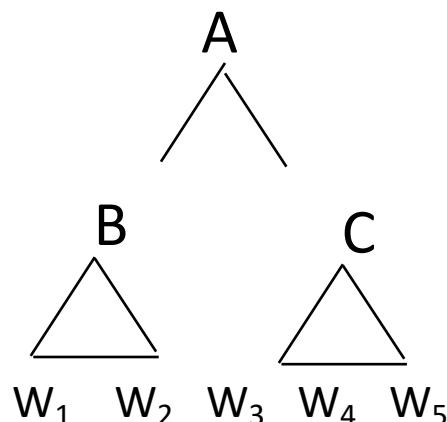
associare probab. a ogni regola d' produzione

$$A \rightarrow BC \quad [p_A]$$

$$P(1,4,A) = p_A * P(1,2,B) * P(3,4,C)$$

$$D \rightarrow BC \quad [p_D]$$

$$P(1,4,D) = p_D * P(1,2,B) * P(3,4,C)$$



Probabilistic CFG

$$G = (\Sigma, V, S, P)$$

$$A \rightarrow \beta [p] \quad p \in (0,1)$$



$$\sum_{\beta} P(A \rightarrow \beta) = 1$$

la somma di tutte le prob.
di $A \rightarrow \beta$ è 1
e' una distribuzione
di probabilità

PCFG

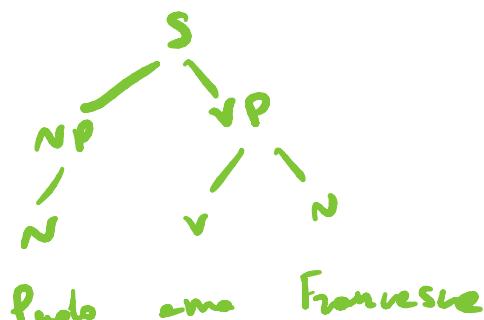
$S \rightarrow NP VP$	$\frac{.80}{.15+}$	{	[.80]	$Det \rightarrow that [.05] \mid the [.80] \mid a [.15]$
$S \rightarrow Aux NP VP$	$\frac{.05}{1}$		[.15]	$Noun \rightarrow book [.10]$
$S \rightarrow VP$			[.05]	$Noun \rightarrow flights [.50]$
$NP \rightarrow Det Nom$			[.20]	$Noun \rightarrow meal [.40]$
$NP \rightarrow Proper-Noun$			[.35]	$Verb \rightarrow book [.30]$
$NP \rightarrow Nom$			[.05]	$Verb \rightarrow include [.30]$
$NP \rightarrow Pronoun$			[.40]	$Verb \rightarrow want [.40]$
$Nom \rightarrow Noun$			[.75]	$Aux \rightarrow can [.40]$
$Nom \rightarrow Noun Nom$			[.20]	$Aux \rightarrow does [.30]$
$Nom \rightarrow Proper-Noun Nom$			[.05]	$Aux \rightarrow do [.30]$
$VP \rightarrow Verb$			[.55]	$Proper-Noun \rightarrow TWA [.40]$
$VP \rightarrow Verb NP$			[.40]	$Proper-Noun \rightarrow Denver [.40]$
$VP \rightarrow Verb NP NP$			[.05]	$Pronoun \rightarrow you [.40] \mid I [.60]$

Modello probabilistico: ipotesi

La probabilità di un albero è il prodotto delle regole usate nella sua derivazione

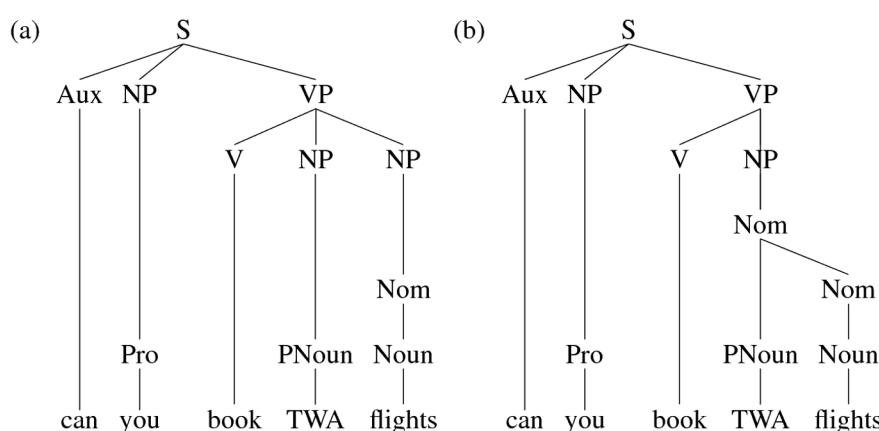
albero
↓
regole

$$P(T, S) = \prod_{node \in T} P(rule(n))$$



↑
dobbiamo recuperare le regole che generano l'albero
le riceviamo livello ~ livello
(perché è una CF)

PCFG



Rules	P	Rules	P
S → Aux NP VP	.15	S → Aux NP VP	.15
NP → Pro	.40	NP → Pro	.40
VP → V NP NP	.05	VP → V NP	.40
NP → Nom	.05	NP → Nom	.05
NP → PNoun	.35	Nom → PNoun Nom	.05
Nom → Noun	.75	Nom → Noun	.75
Aux → Can	.40	Aux → Can	.40
NP → Pro	.40	NP → Pro	.40
Pro → you	.40	Pro → you	.40
Verb → book	.30	Verb → book	.30
PNoun → TWA	.40	PNoun → TWA	.40
Noun → flights	.50	Noun → flights	.50

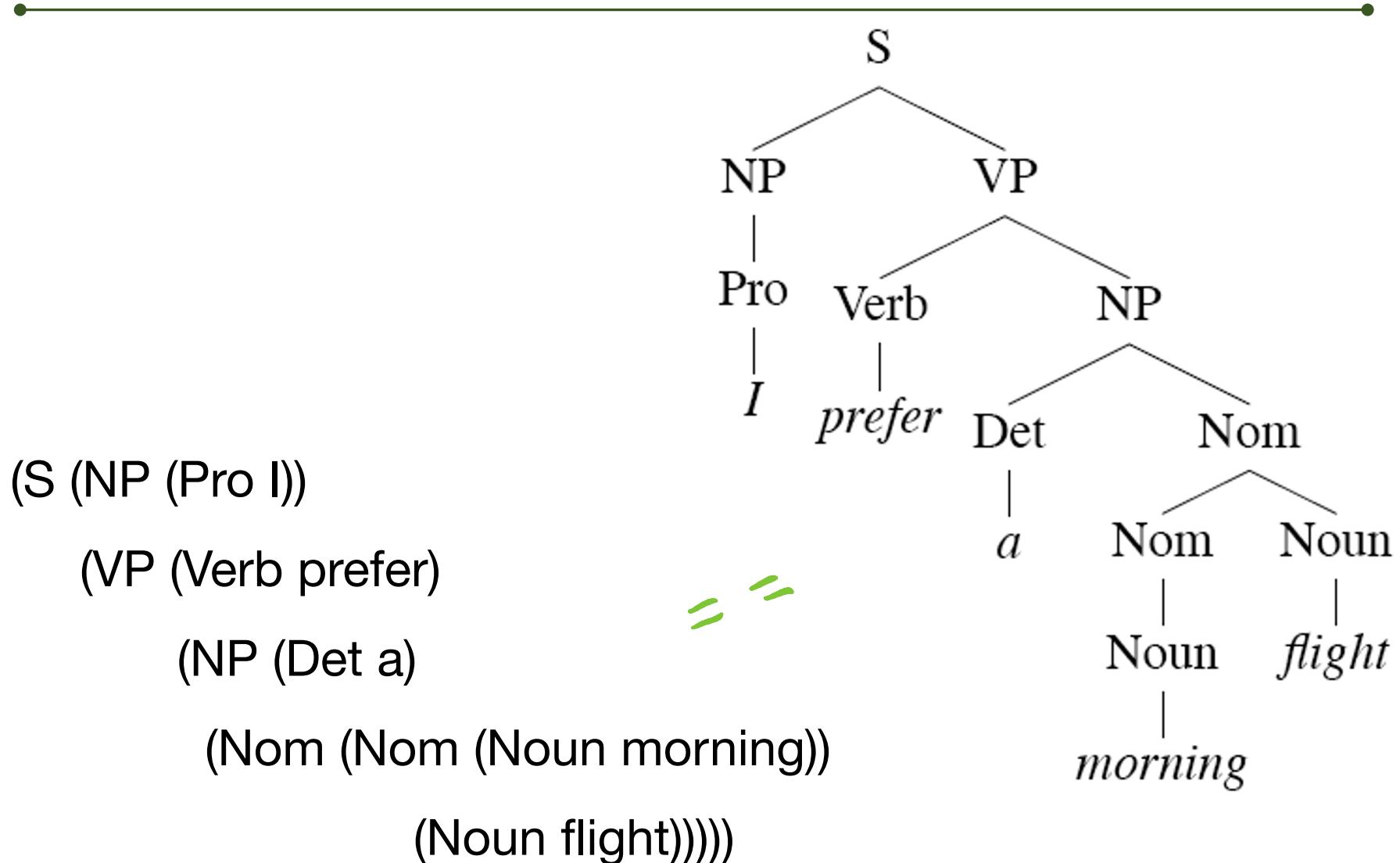
$$P(T_a) = .15 * .4 * .05 * .05 * \\ .35 * .75 * .4 * .4 * \\ .4 * .3 * .4 * .5 = \\ = 1.5 \times 10^{-6}$$

$$P(T_b) = .15 * .4 * .05 * \\ .05 * .75 * .4 * .4 * \\ .4 * .3 * .4 * .5 = \\ = 1.7 \times 10^{-6}$$

Da dove prendo le probabilità?

- Treebanks -> banche di alberi *corpus* ← tanti alberi da cui posso ricavare le prob.
- ○ Costituenti -> Penn TB
- ○ Dipendenze -> TUT TB -> Universal dependency TB
- Si creano con sistemi semiautomatici:
 - prima un parser
 - poi correggo a mano
- Guidelines di annotazione
- Corpus linguistics

S-espressioni e alberi



Penn TB

Wall Street Journal

section of the Penn

TreeBank. 1 M words

from the 1987-1989

Wall Street Journal.

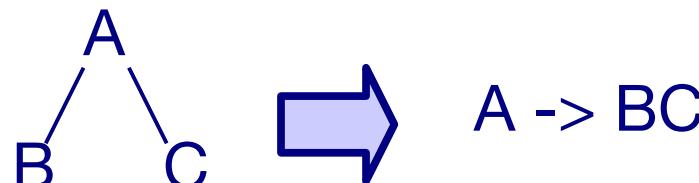
```
( (S ('' '')  
  (S-TPC-2  
    (NP-SBJ-1 (PRP We) )  
    (VP (MD would)  
      (VP (VB have)  
        (S  
          (NP-SBJ (-NONE- *-1) )  
          (VP (TO to)  
            (VP (VB wait)  
              (SBAR-TMP (IN until)  
                (S  
                  (NP-SBJ (PRP we) )  
                  (VP (VBP have)  
                    (VP (VBN collected)  
                      (PP-CLR (IN on)  
                        (NP (DT those)(NNS assets)))))))))))  
                (, ,) ('' '')  
                (NP-SBJ (PRP he) )  
                (VP (VBD said)  
                  (S (-NONE- *T*-2) ))  
                ( . .) )))
```

TB grammars

- I TB implicitamente definiscono delle grammatiche CF
 - induzione (?)

- Regole locali:

parte del TB



- WSJ -> 12k regole (!!)

- A volte regole flat, gli annotatori evitano la ricorsione:

VP → VBD PP
VP → VBD PP PP
VP → VBD PP PP PP
VP → VBD PP PP PP PP

Modello probabilistico: training

- Abbiamo bisogno di

$$P(\alpha \rightarrow \beta | \alpha)$$

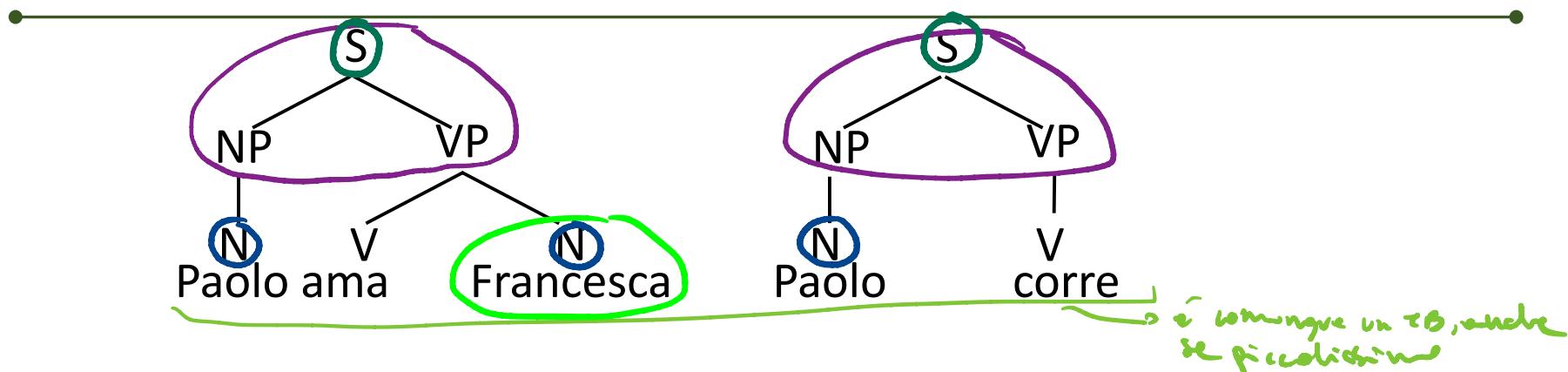
- Quindi, per ogni albero nel TB, calcoliamo:

$$P(\alpha \rightarrow \beta | \alpha) = \frac{\text{Count}(\alpha \rightarrow \beta)}{\sum_{\gamma} \text{Count}(\alpha \rightarrow \gamma)} = \frac{\text{Count}(\alpha \rightarrow \beta)}{\text{Count}(\alpha)}$$

→
risponiglie
all'H&N

↗ # di appariz. di una regola nel TB
↗ # di appariz. della testa
nel TB

Treebank Grammars (PCFG)



$$P(S \rightarrow NP\ VP) = 2/2 = 1$$

$$P(NP \rightarrow N) = 2/2 = 1$$

$$P(VP \rightarrow V\ N) = 1/2 = .5$$

$$P(VP \rightarrow V) = 1/2 = .5$$

$$P(N \rightarrow \text{Paolo}) = 2/3 = .66$$

$$P(N \rightarrow \text{Francesca}) = 1/3 = .33$$

$$P(V \rightarrow \text{corre}) = 1/2 = .5$$

$$P(V \rightarrow \text{ama}) = 1/2 = .5$$

Parser C: CKY probabilistico

function PROBABILISTIC-CKY(*words, grammar*) **returns** most probable parse
and its probability

```
for  $j \leftarrow$  from 1 to LENGTH(words) do
    for all {  $A \mid A \rightarrow words[j] \in grammar$  }
        table[ $j - 1, j, A$ ]  $\leftarrow P(A \rightarrow words[j])$ 
for  $i \leftarrow$  from  $j - 2$  downto 0 do
    for  $k \leftarrow i + 1$  to  $j - 1$  do
        for all {  $A \mid A \rightarrow BC \in grammar$ ,
                    and table[ $i, k, B$ ]  $> 0$  and table[ $k, j, C$ ]  $> 0$  }
            if (table[ $i, j, A$ ]  $<$   $P(A \rightarrow BC) \times table[i, k, B] \times table[k, j, C]$ ) then
                table[ $i, j, A$ ]  $\leftarrow P(A \rightarrow BC) \times table[i, k, B] \times table[k, j, C]$ 
                back[ $i, j, A$ ]  $\leftarrow \{k, B, C\}$ 
return BUILD-TREE(back[1, LENGTH(words), S]), table[1, LENGTH(words), S]
```

è pseudo codice: oltre al avremo
(la prob. + altri) manterò anche
il back pointer al sotto - alberi

se il know è +
prob. lo sostituisce
al vecchio

us CKY senza
prob - dove
c'era una
Union

PRO: - rimuove un po' di ambiguità \rightarrow esp. comb.
CON: - è un po' + lento

è un beam-search

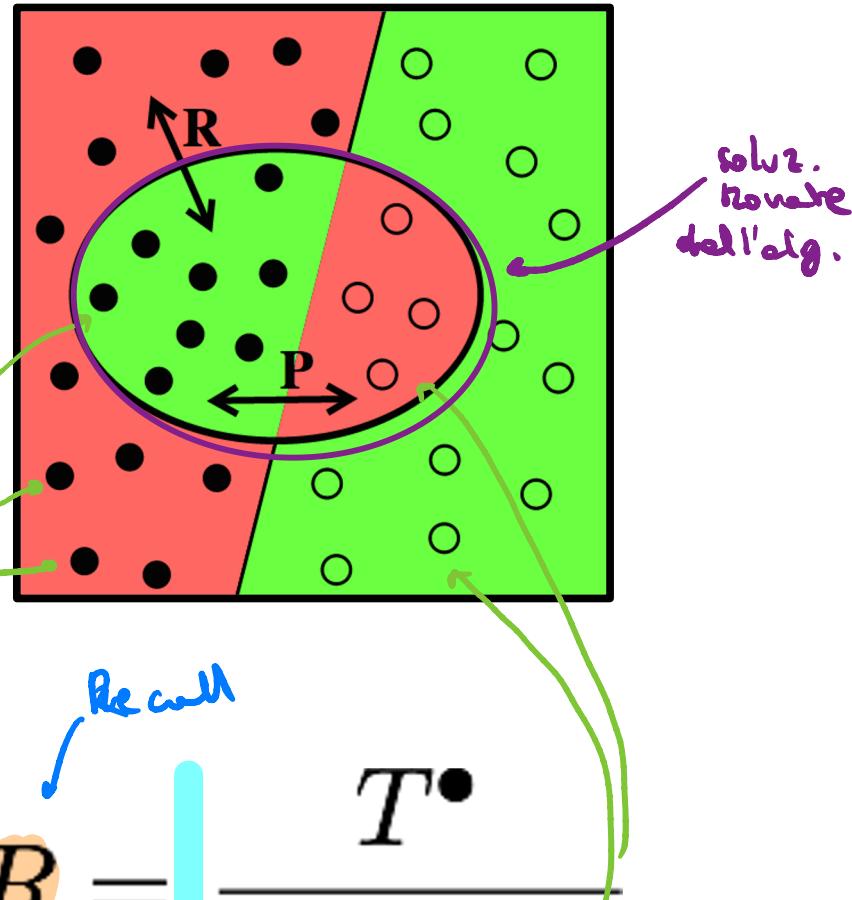
Tecnologie del Linguaggio Naturale - Informatica - 2020/2021

nella tabella ora
metto delle distrib.
di prob.

Valutazione

- Accuracy
- Precision vs. Recall
- F-score
- Unlabelled vs. Labelled

quanto è buono un albero di output del cky



Precision

$$P = \frac{T^\bullet}{T^\bullet + F^\bullet}$$

T[•] true
F[•] false

Recall

$$R = \frac{T^\bullet}{T^\bullet + F^\circ}$$

Valutazione per le PCFG

- Precision
 - Quale percentuale di subtree del system tree sono anche nel gold tree? >
 - Quanto di quello prodotto è giusto?
- Recall
 - Quale percentuale di subtree del golden tree sono anche nel system tree? ->
 - Quanto di quello che avremmo dovuto produrre abbiamo prodotto?

Parseval

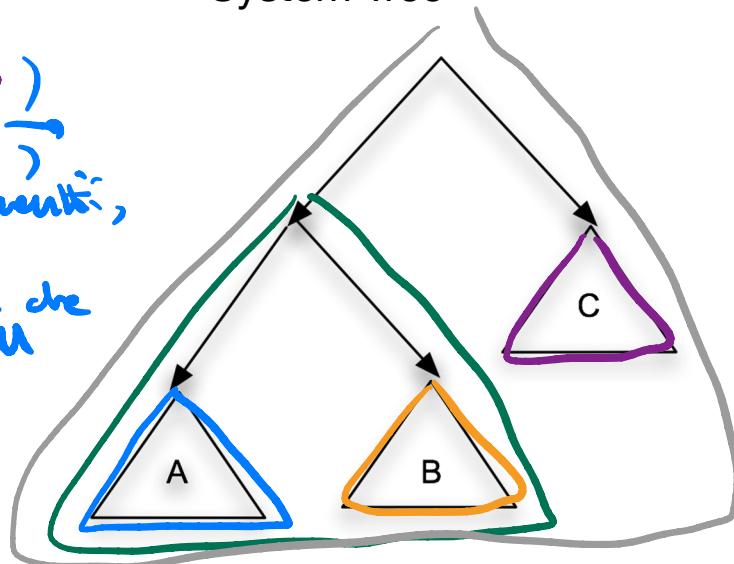
partition del
TB

- TRAINING
- DEVELOPMENT (VALIDATION)
- TEST SET

- Crossing brackets

albero ottenuto dal
parser

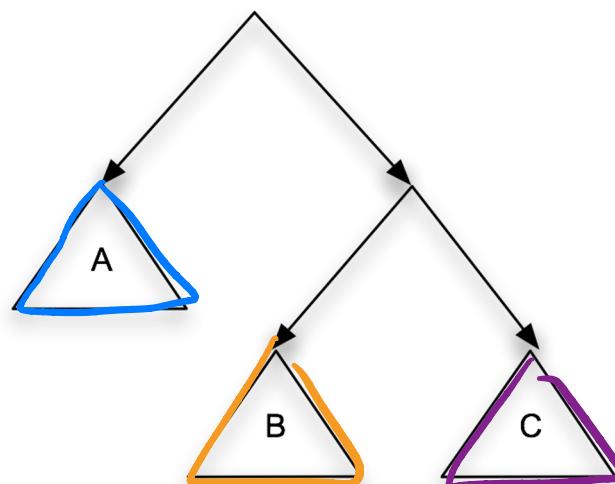
System Tree



((A B) C)

albero originale
proveniente dal
sistema ("quello che avrebbe
detto il linguista")

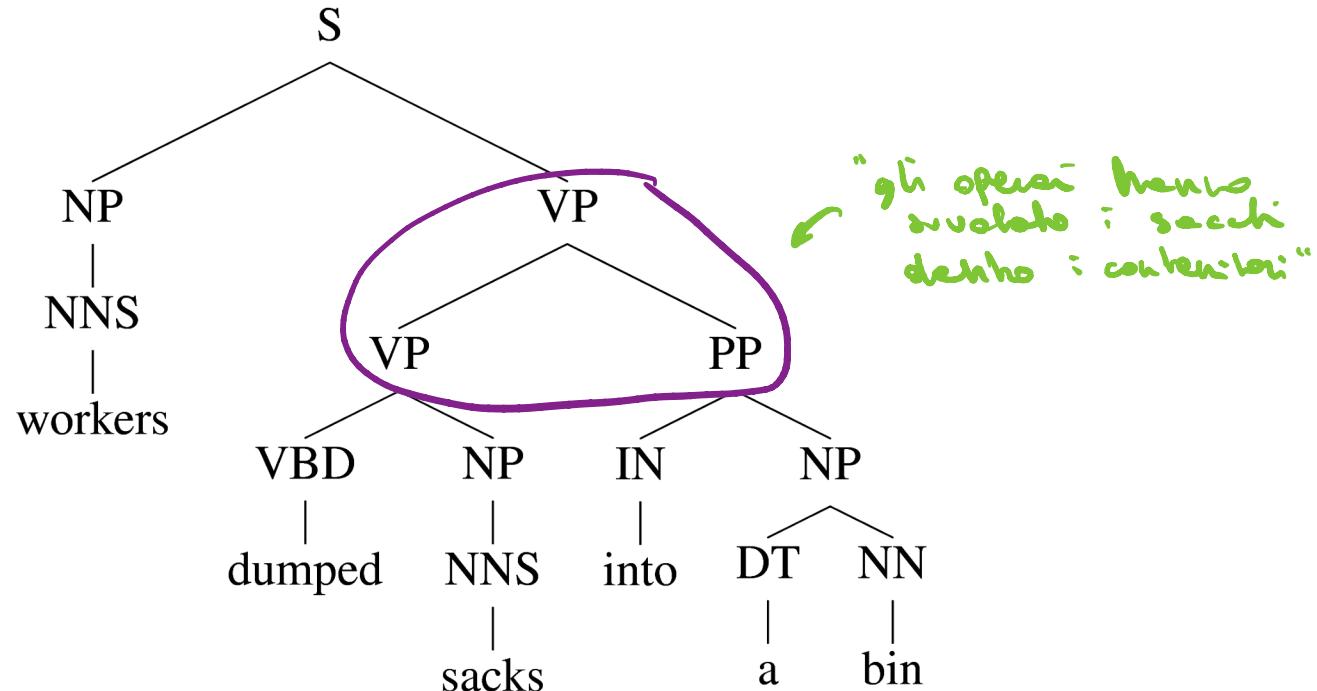
Gold Tree



(A (B C))

Problem with PCFG

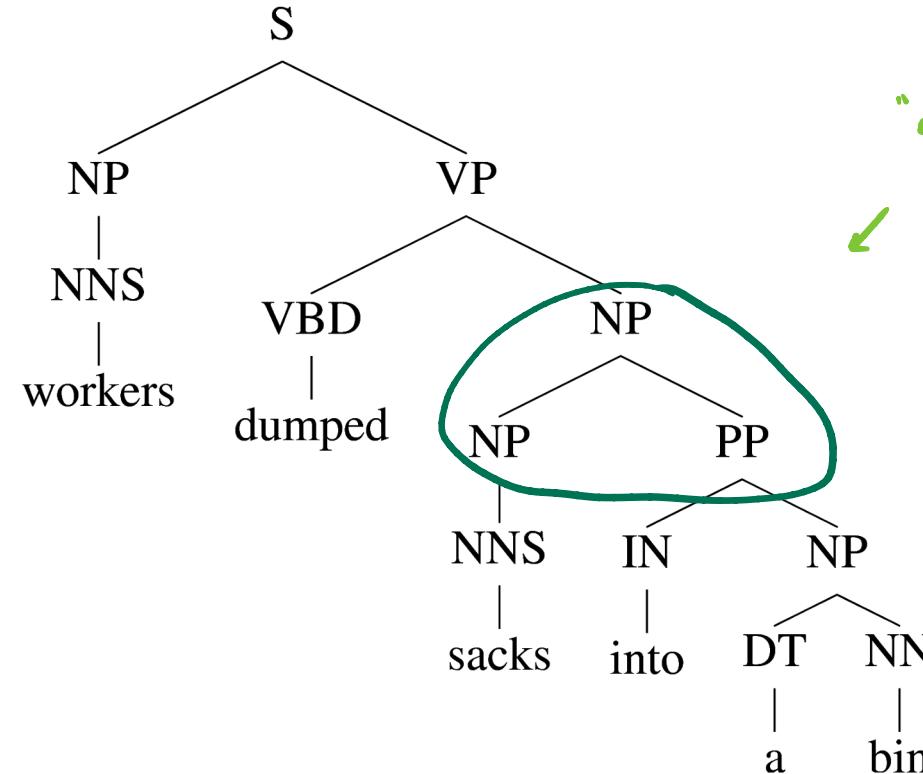
(a)



Ipotesi di indipendenza -> *non ci sono preferenze lessicali*

Problem with PCFG

(b)



"gli operai hanno
svuotato i sacchi
che erano dentro
i contenitori"

del punto di
viste delle
sintassi, queste sono
e quelle sopra sono
molto simili, e
meno di o e o,
mentre io vorrei
che ci fossero delle
differenze in prob.
(che una parola +
prob. dell'altra).

Quindi,

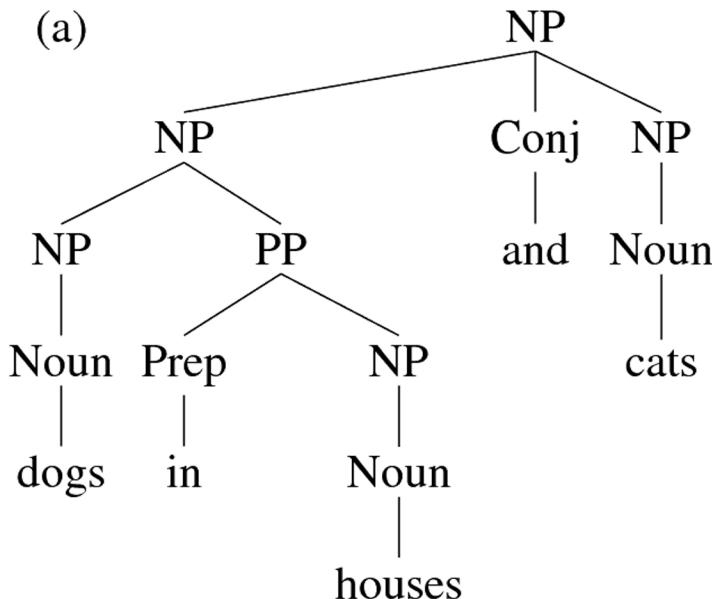
Ipotesi di indipendenza -> *non ci sono preferenze lessicali*

Problem with PCFG

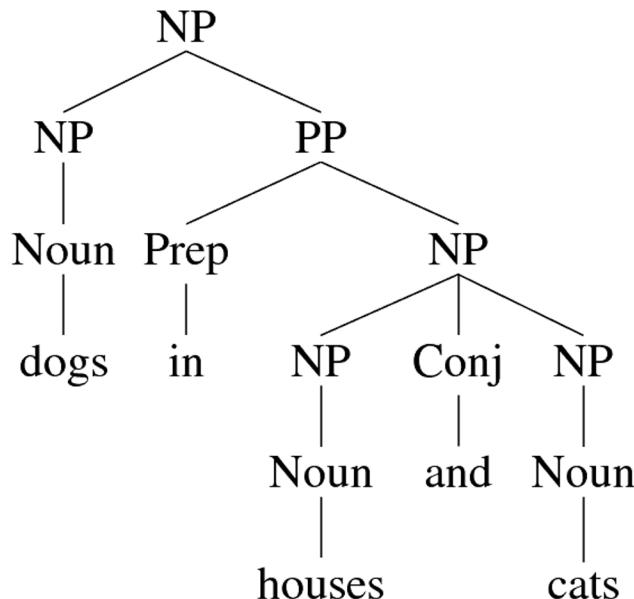
- 9 anche questi 2 alberi, che hanno una semantica molto diversa, hanno le stesse prob. dal punto di vista sintattico (mentre vorremmo che fossero diversi)



(a)



(b)



Ipotesi di indipendenza -> *non ci sono preferenze lessicali*

Lexicalized PCFG

Ogni regola CF è potenziata con informazione

circa le heads dei costituenti coinvolti

$A \rightarrow BC$

$A(\text{head}_A) \rightarrow B(\text{head}_B) C(\text{head}_C)$

Punto di unione tra formalismi sintattici

a costituenti e a dipendenza

Lexicalized PCFG

uso regole linguistiche x analisi
- ogni nodo un symb. NT, ma anche
qual è il remin. nella frase del TB
che ha + influito nella costruzione di
quel NT

VP → VBD NP PP



VP(dumped) → VBD(dumped) NP(sacks) PP(into)
 $[3 \times 10^{-10}]$

VP(dumped) → VBD(dumped) NP(cats) PP(into)
 $[8 \times 10^{-11}]$

VP(dumped) → VBD(dumped) NP(hats) PP(into)
 $[4 \times 10^{-10}]$

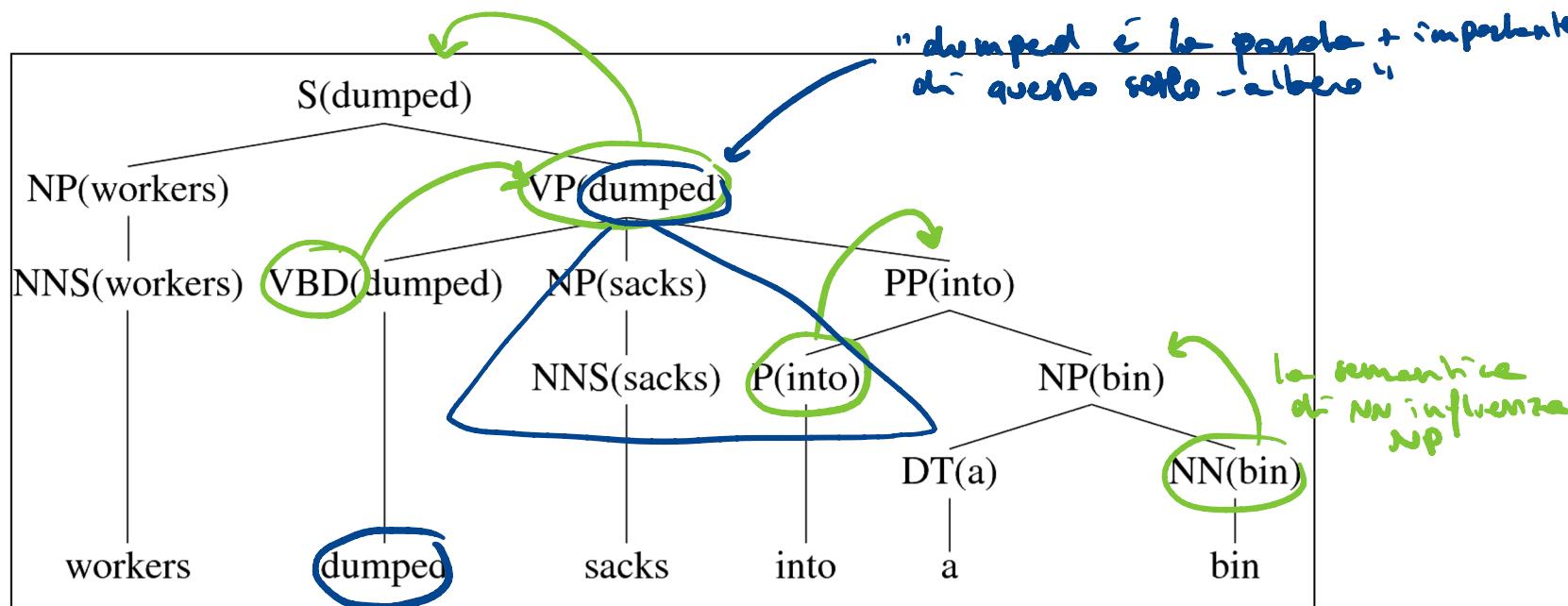
VP(dumped) → VBD(dumped) NP(sacks) PP(above)
 $[1 \times 10^{-12}]$

delle stesse regole ora ne ho 6

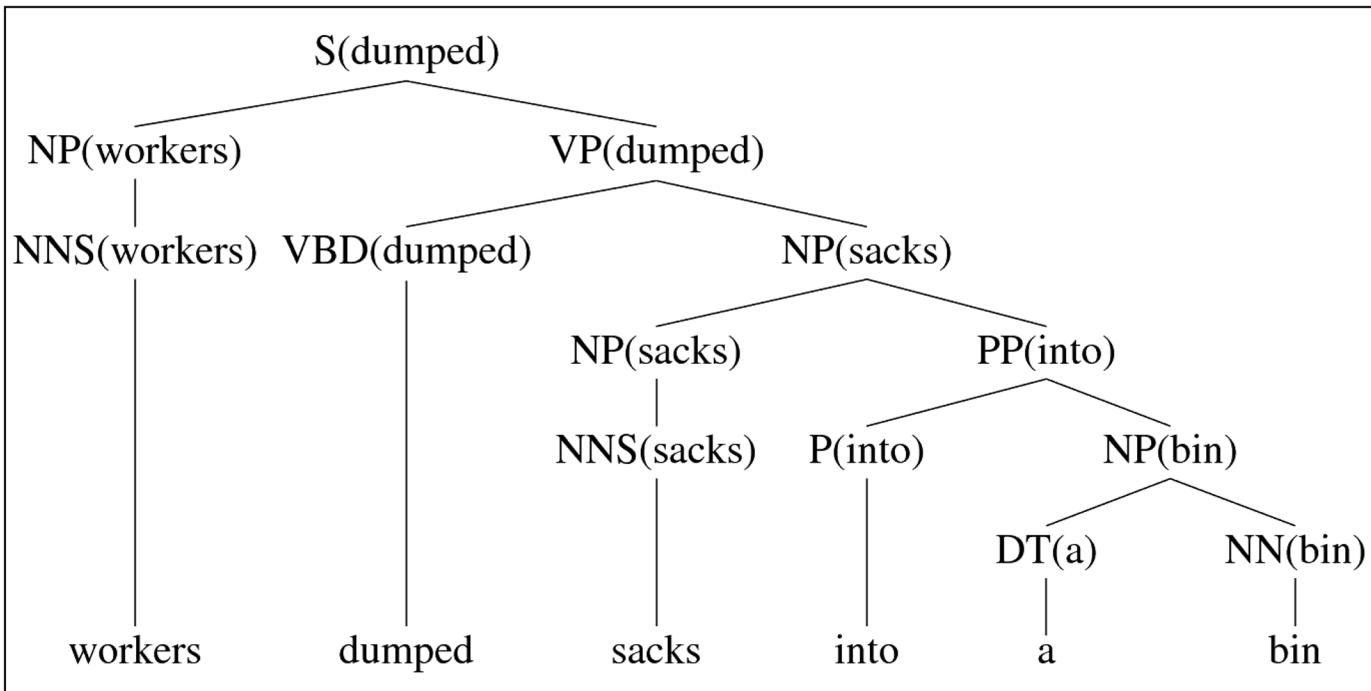
PERCOLAZIONE : faccio emergere l'elemento
lesicale + imparare al genitore

così il numero di regole esplode => deep wene una tecnica di smoothing
ma l'ho rifatto

Lexicalized PCFG



Lexicalized PCFG



Outline

- Parser anatomy
- Top-down vs. bottom-up
- CKY algorithm: Cocke-Kasami-Younger
- Parsing probabilistico e TB grammars
- ~~Parsing parziale~~
*non lo vediamo
x questioni di tempo!*

Parser D (Chunk parsing a regole)

(1) Grammar

Regular-Grammars (cascade), ...

(2) Algorithm

I. Search strategy

top-down, bottom-up, left-to-right, ...

II. Memory organization

back-tracking, dynamic programming, ...

(3) Oracle

Probabilistic, rule-based, ...

Parsing parziale -> chunk parsing

- Base syntactic units-> CHUNK
 - chuncks=basic non-recursive phrases
- Elimino la ricorsione!

[*NP* The morning flight] [*PP* from] [*NP* Denver] [*VP* has arrived.]

[*NP* a flight] [*PP* from] [*NP* Indianapolis][*PP* to][*NP* Houston][*PP* on][*NP* TWA]

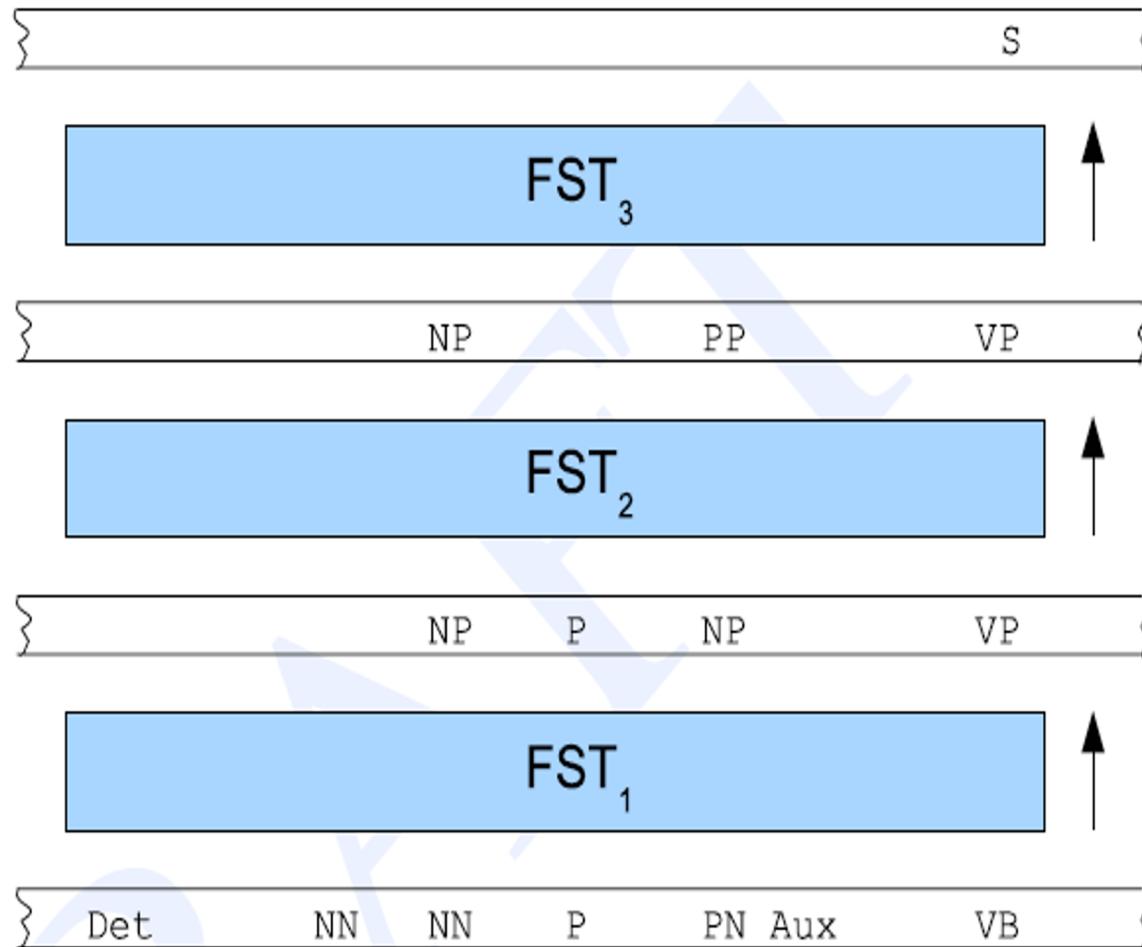
[*NP* The morning flight] from [*NP* Denver] has arrived.

Chunk parsing a regole

- Escludo la ricorsione: i sintagmi possono essere ricorsivi i chunk **NO**
- Cascata di regole gerarchiche

$$NP \rightarrow (Det) Noun^* Noun$$
$$NP \rightarrow Proper-Noun$$
$$VP \rightarrow Verb$$
$$VP \rightarrow Aux Verb$$

Cascata di trasduttori: • approssimo una CFG



The morning flight from Denver has arrived

Parser E (Chunk parsing probabilistico)

(1) Grammar

Regular-Grammars (cascade), ...

(2) Algorithm

I. Search strategy

top-down, bottom-up, left-to-right, ...

II. Memory organization

back-tracking, greedy (depth first), ...

(3) Oracle

Probabilistic, rule-based, ...

IOB tagging

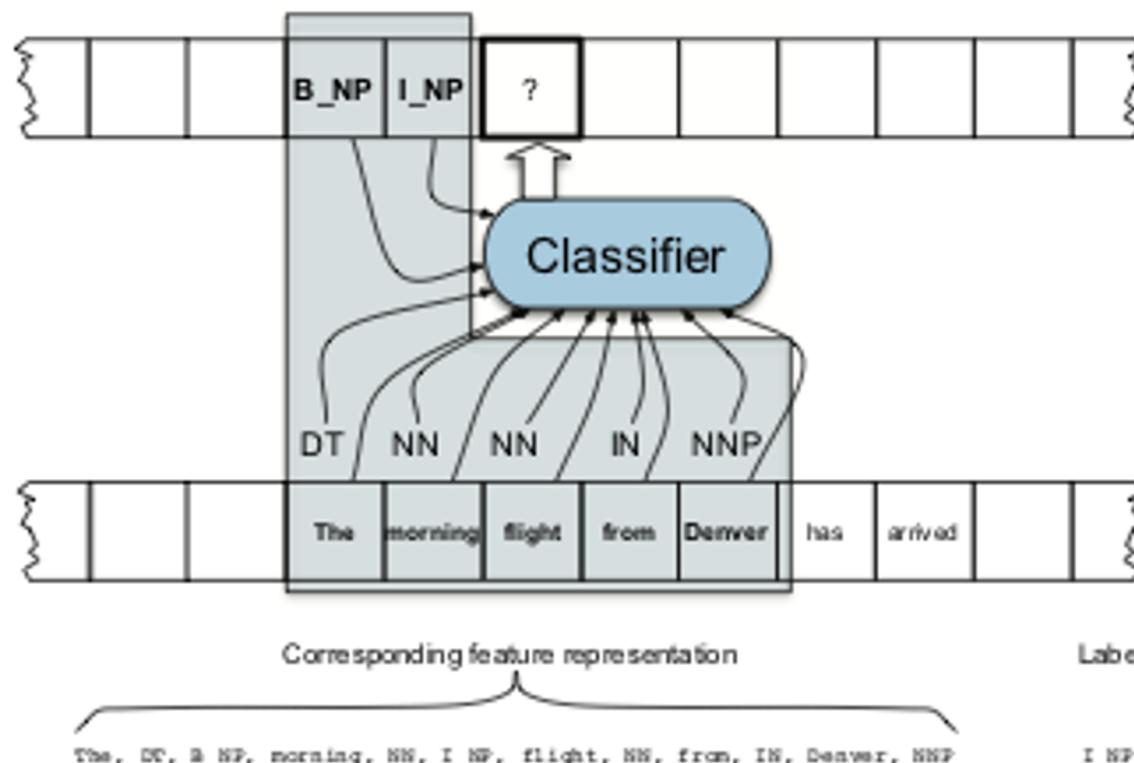
- Chunck parsing -> tagging
- n chuncks -> $2n+1$ tags
- **B**egin , **I**nside, **O**utside

The morning flight from Denver has arrived
B_NP I_NP I_NP B_PP B_NP B_VP I_VP

The morning flight from Denver has arrived.
B_NP I_NP I_NP O B_NP O O

IOB tagging: learning

- Chunk parsing -> tagging
- n chunks -> $2n+1$ tags
- Begin , Inside, Outside



E se le CFG non ci piacciono?

Back in U.S.S.R.

-> grammatiche a dipendenze