premessa: si è scelta la strada NL ≅ CF

Tecnologie del Linguaggio Naturale

Parte Prima

Lezione n. 04-2

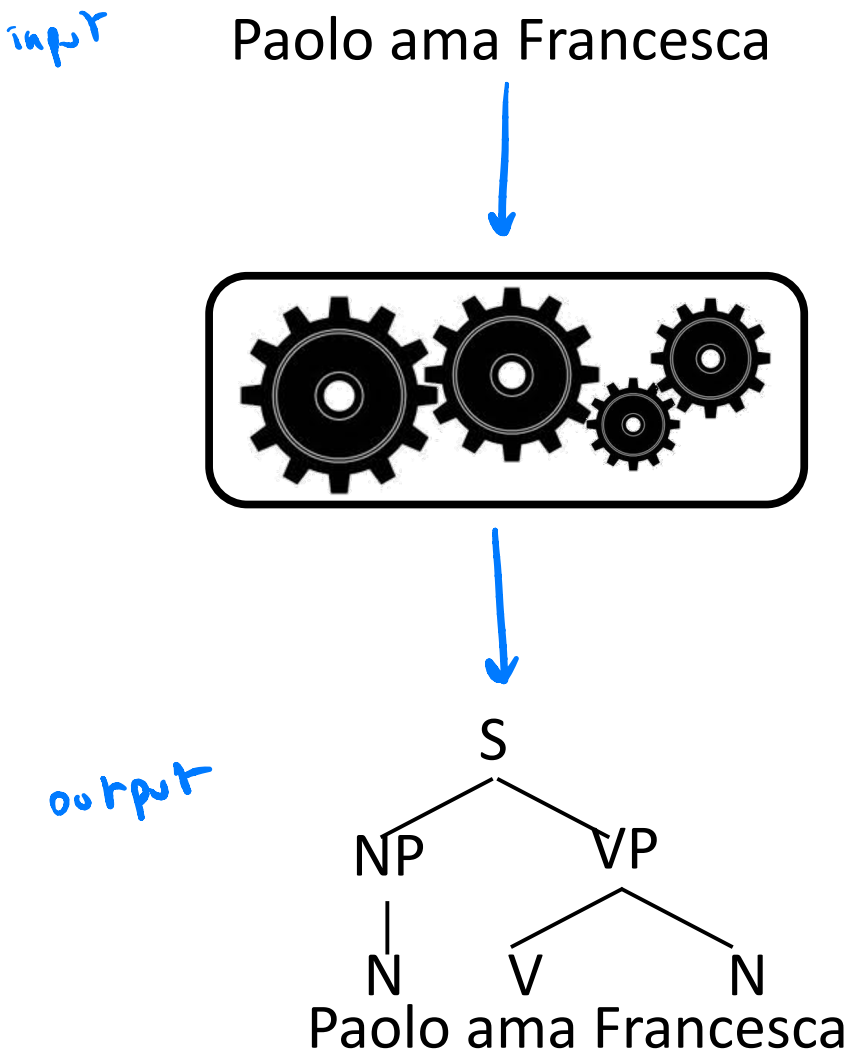
Sintassi: la performance (a costituenti)

15-03-2021

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

↑
quali algo. usare x ottenerla?

Parser


input
Paolo ama Francesca
output
S
NP
VP
N
V
N
Paolo ama Francesca

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

(1) Grammar (or automa model)

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

(2) Algorithm

I. Search strategy

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

II. Memory organization

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

(3) Oracle

Probabilistic, rule-based, ...

Outline

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

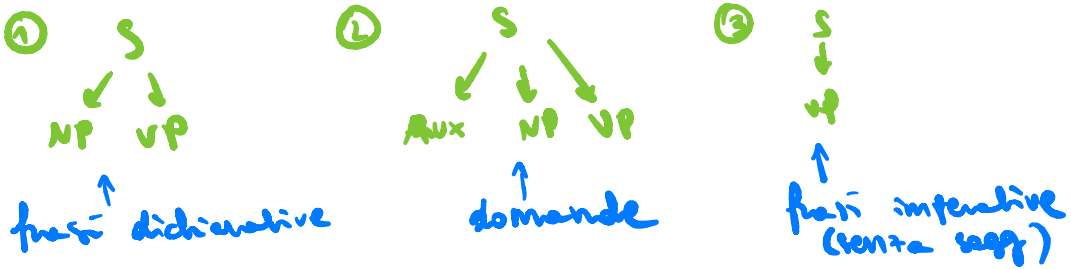
Grammar

sono fondamentali: i PoS ← questa grammatica definisce anche i PoS tagging. Potrebbe non essere così

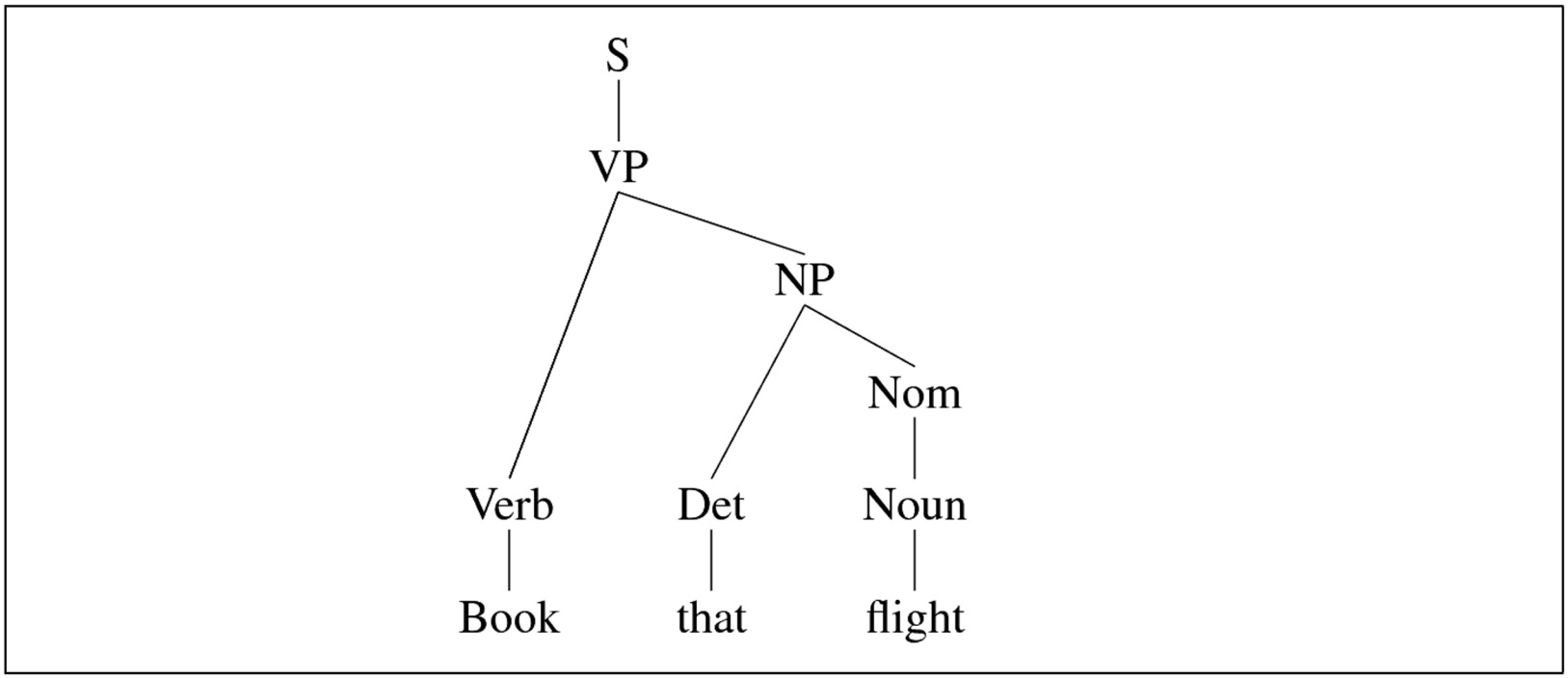
1 $S \rightarrow NP\ VP$	$Det \rightarrow that \mid this \mid a$
2 $S \rightarrow Aux\ NP\ VP$	$Noun \rightarrow book \mid flight \mid meal \mid money$
3 $S \rightarrow VP$	$Verb \rightarrow book \mid include \mid prefer$
$NP \rightarrow Det\ Nominal$	$Aux \rightarrow does$
$Nominal \rightarrow Noun$	
$Nominal \rightarrow Noun\ Nominal$	$Prep \rightarrow from \mid to \mid on$
$NP \rightarrow Proper\text{-}Noun$	$Proper\text{-}Noun \rightarrow Houston \mid TWA$
$VP \rightarrow Verb$	
$VP \rightarrow Verb\ NP$	$Nominal \rightarrow Nominal\ PP$

gestendo a questo livello il PoS tagging non riesco a eliminare le ambiguità: book è verbo o nome?

come fa bene il PoS tagger



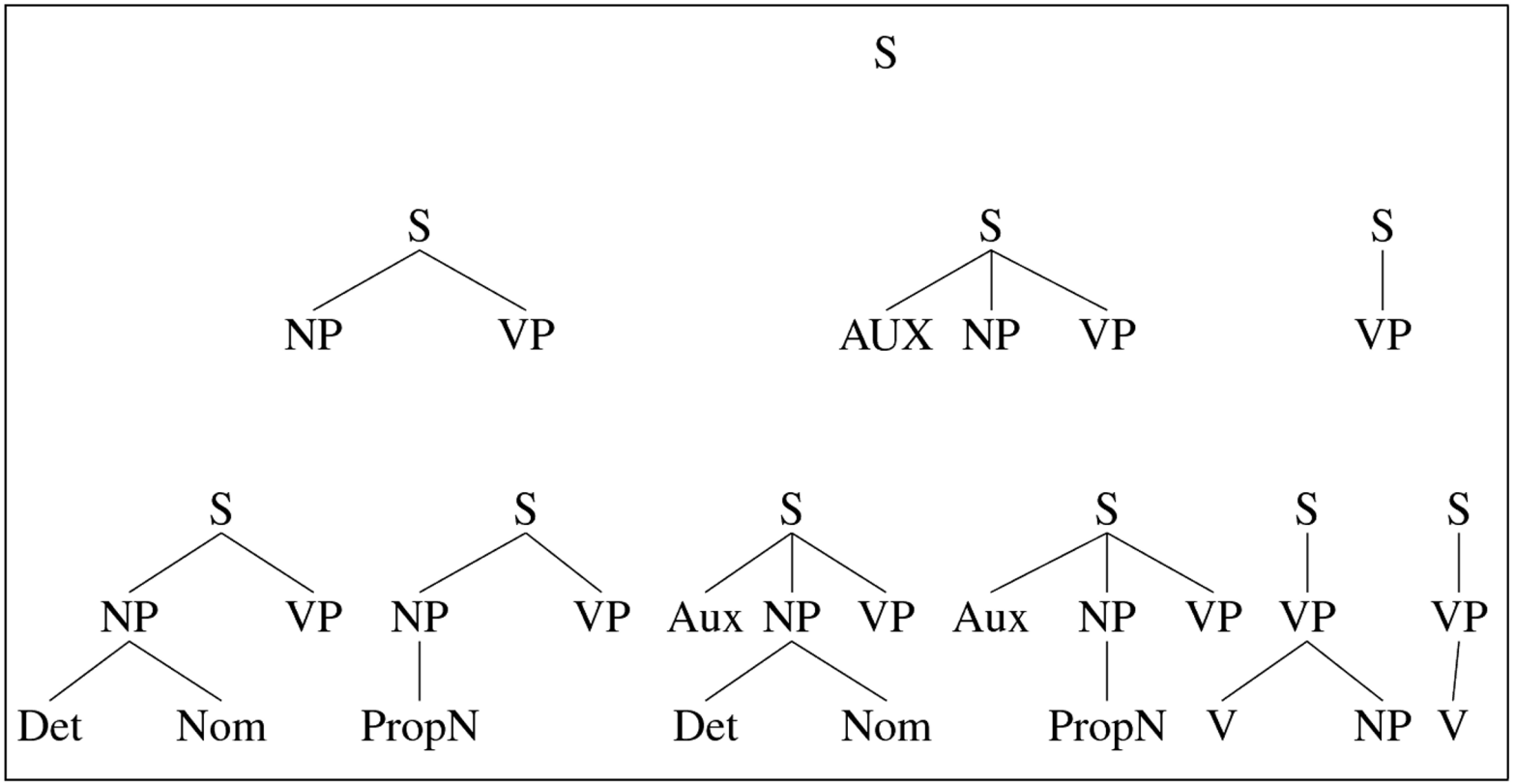
Target Parse



Information sources

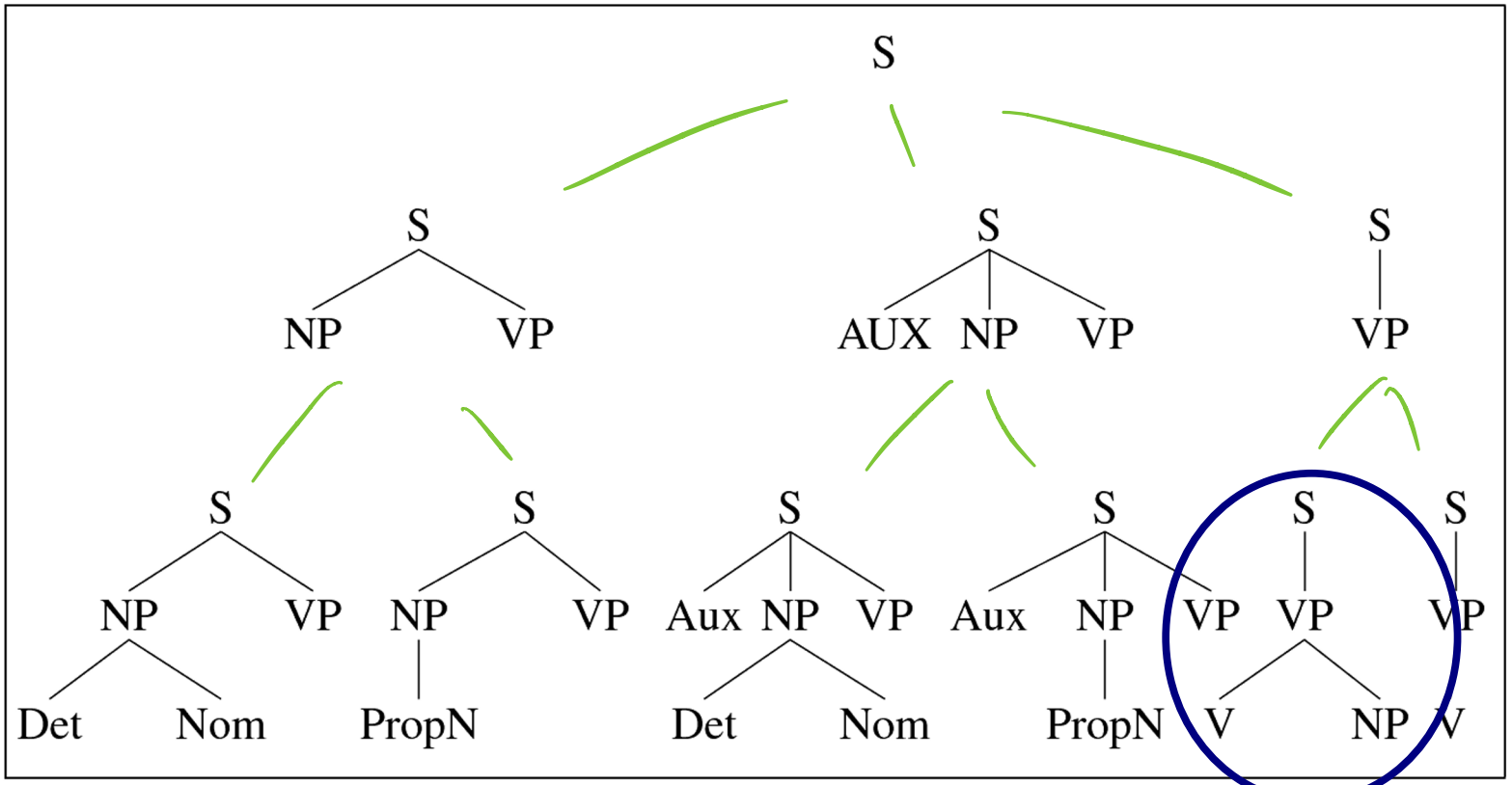
- The grammar -> goal-directed parsing -> top-down
  - Solo ricerche che possono portare a risposte corrette, cioè con radice S
  - Ma comporta la creazione di alberi non compatibili con le parole
  - Razionalisti
- The sentence -> data-directed parsing -> bottom-up
  - Solo ricerche compatabili con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

Top-Down



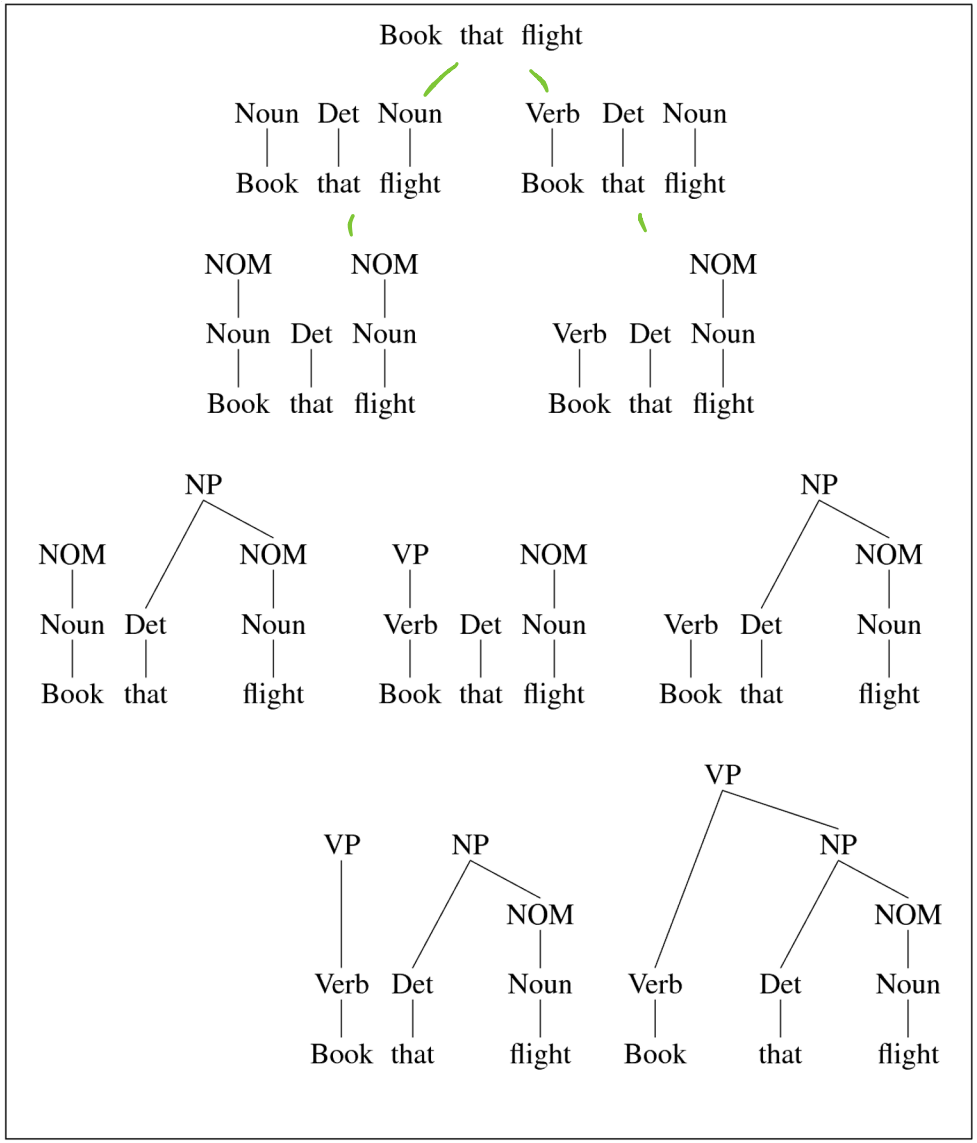
Book that flight

Top-Down

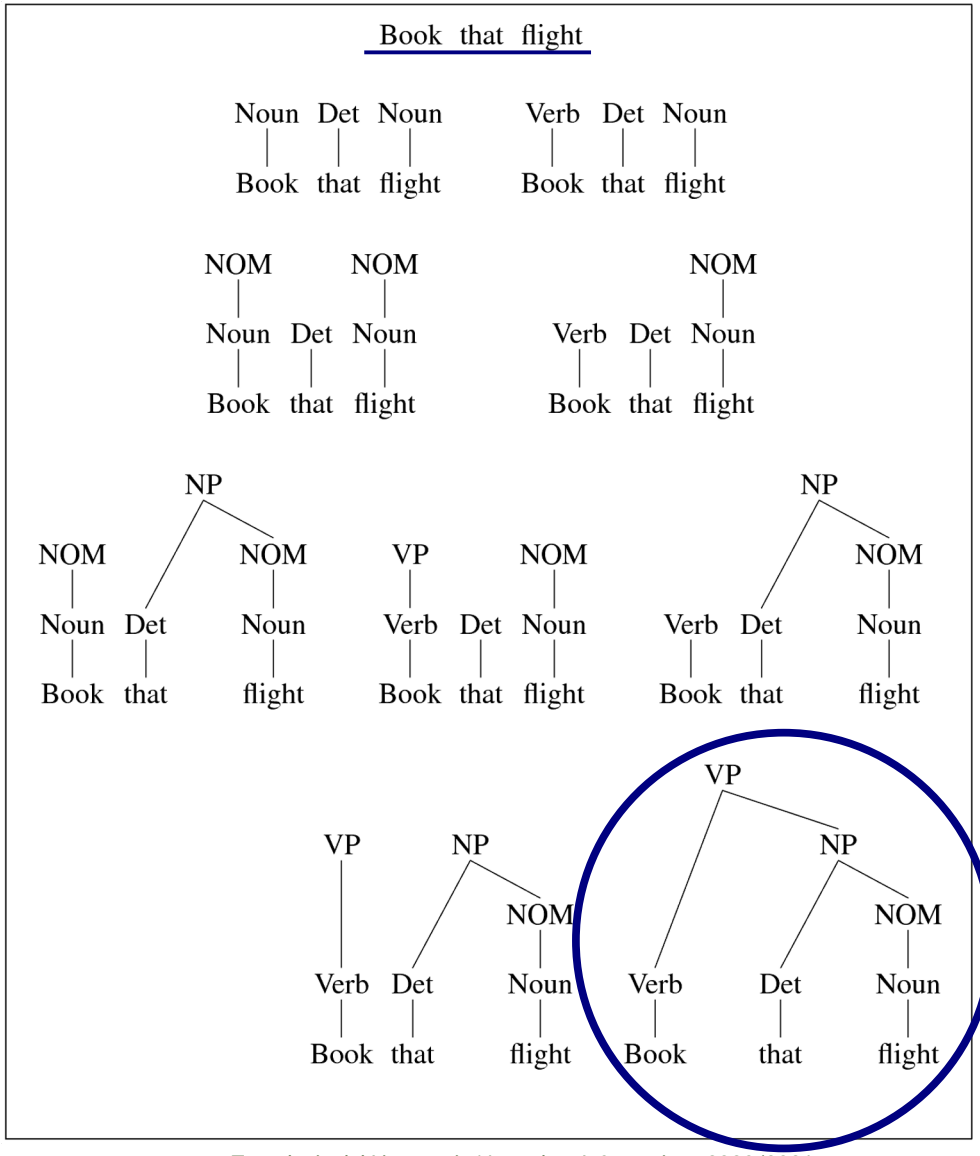


Book that flight

Bottom-Up


Book that flight
Noun Det Noun
Book that flight
Verb Det Noun
Book that flight
NOM NOM
Noun Det Noun
Book that flight
NOM
Verb Det Noun
Book that flight
NP
NOM NOM
Noun Det Noun
Book that flight
VP NOM
Verb Det Noun
Book that flight
NP
NOM
Verb Det Noun
Book that flight
VP
VP NP
NOM
Verb Det Noun
Book that flight
VP
NP
NP
NOM
Verb Det Noun
Book that flight

Bottom-Up


Book that flight
Noun Det Noun
Book that flight
Verb Det Noun
Book that flight
NOM NOM
Noun Det Noun
Book that flight
NOM
Verb Det Noun
Book that flight
NP
NOM NOM
Noun Det Noun
Book that flight
VP NOM
Verb Det Noun
Book that flight
NP
NOM
Verb Det Noun
Book that flight
VP
VP NP
NOM
Verb Det Noun
Book that flight
VP
NP NP
Verb Det NOM Verb Det NOM
Book that flight Book that flight

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, ...

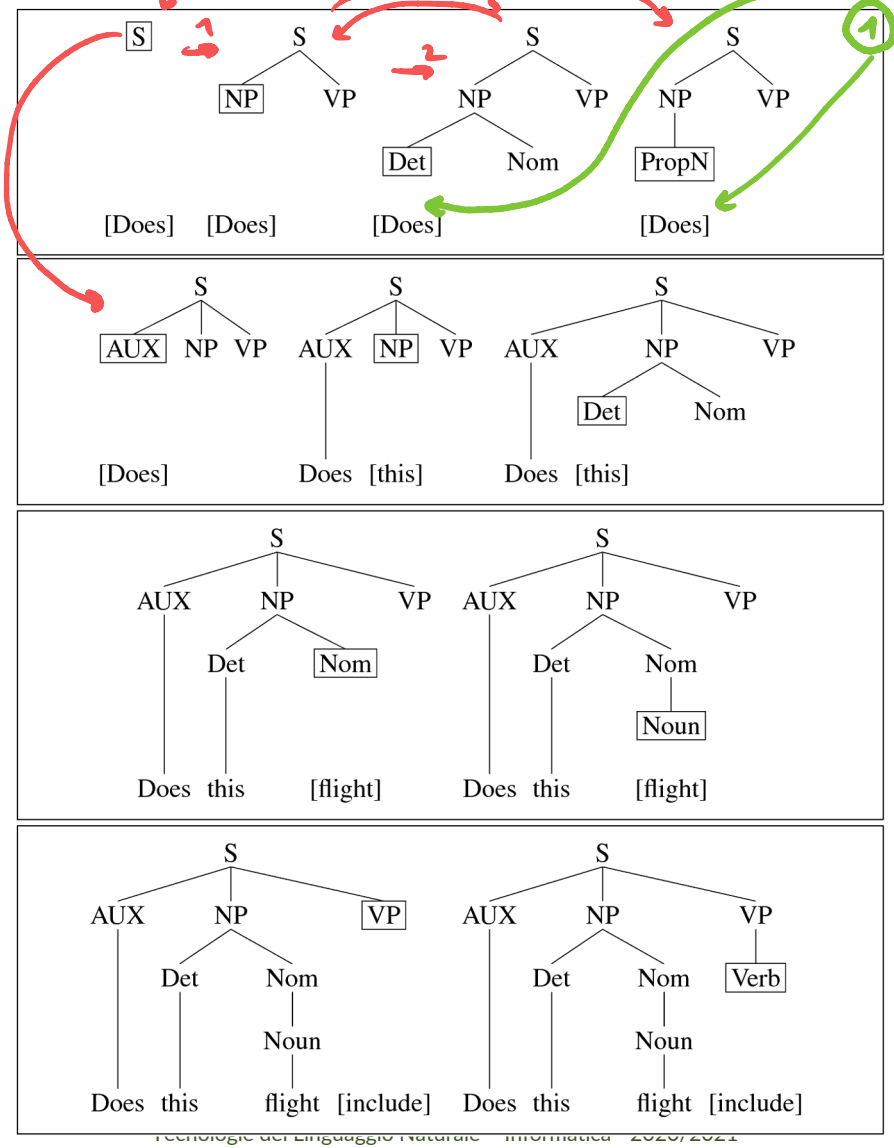
(3) Oracle

Probabilistic, rule-based, ...

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

Parser A - 1

Does this flight include a meal …



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

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

- $Prep \rightarrow from \mid to \mid on$
- $Proper\text{-}Noun \rightarrow Houston \mid TWA$

- $Nominal \rightarrow Nominal\ PP$

S→NP VP
NP→DET Nom
NP→Prop N

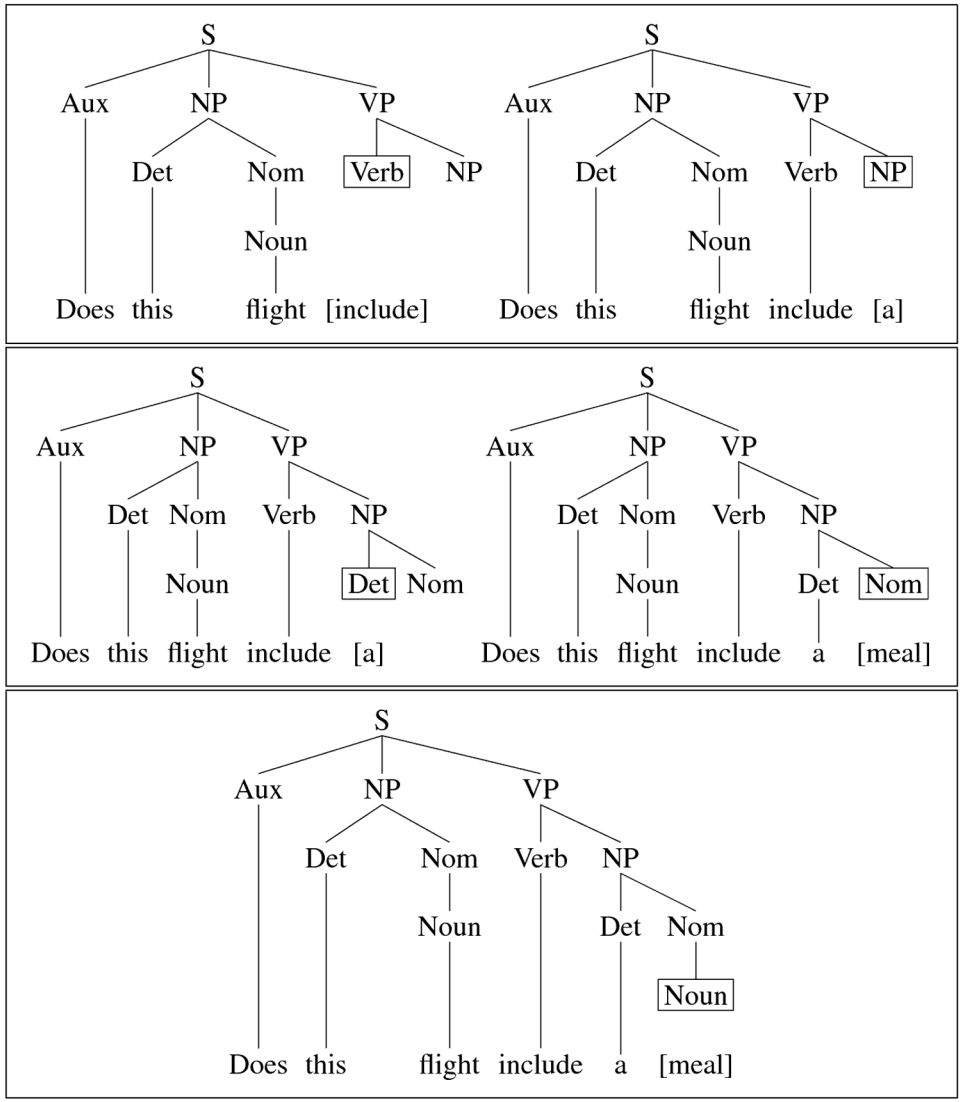
S→AUX NP VP
AUX→does
NP→DET Nom

DET→this
Nom→Noun

Noun→flight
VP→Verb

Parser A - 2

si arriva a una soluzione? sì, ma: con 1 left-recursion 2 shared sub-problems



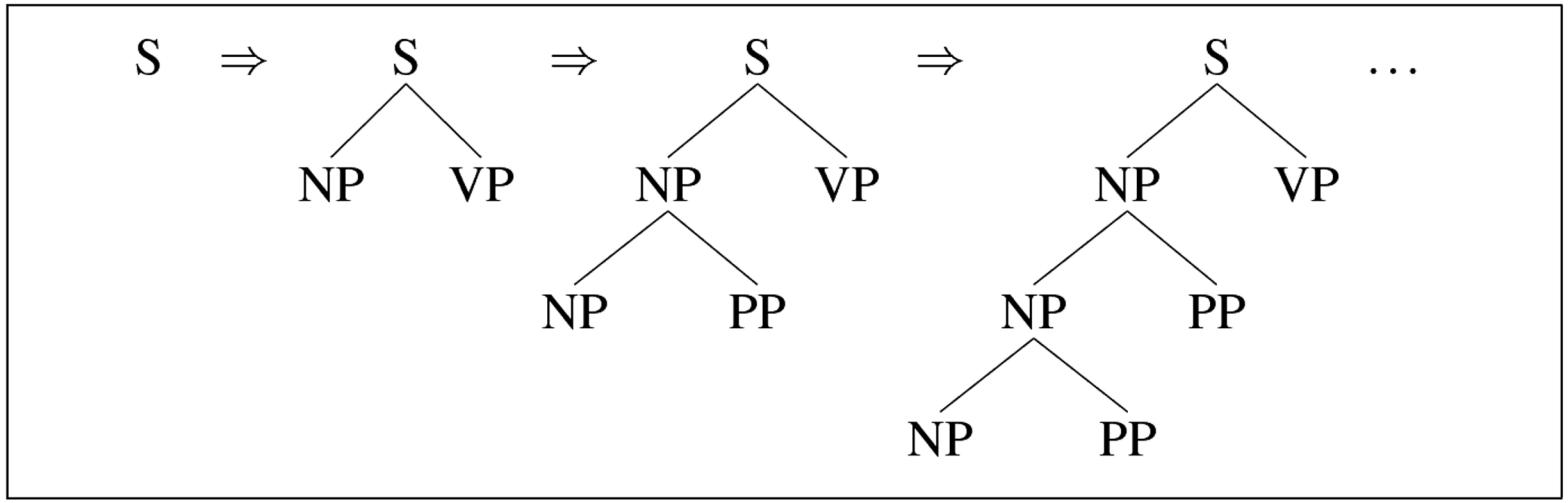
VP→Verb NP
Verb→include

NP→Det Nom
Det→a

Nom→Noun
Noun→meal

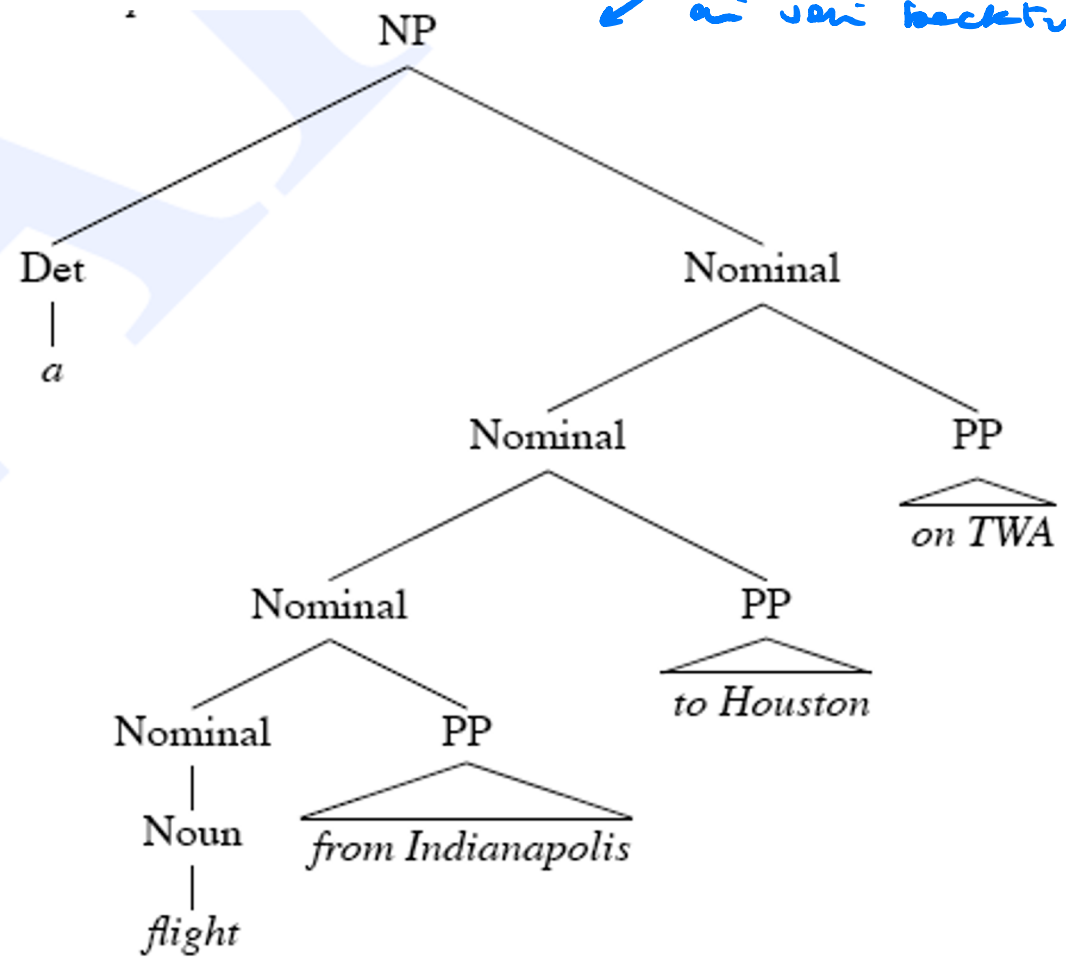
① Left-Recursion

$$NP \rightarrow NP\ PP$$



va in loop

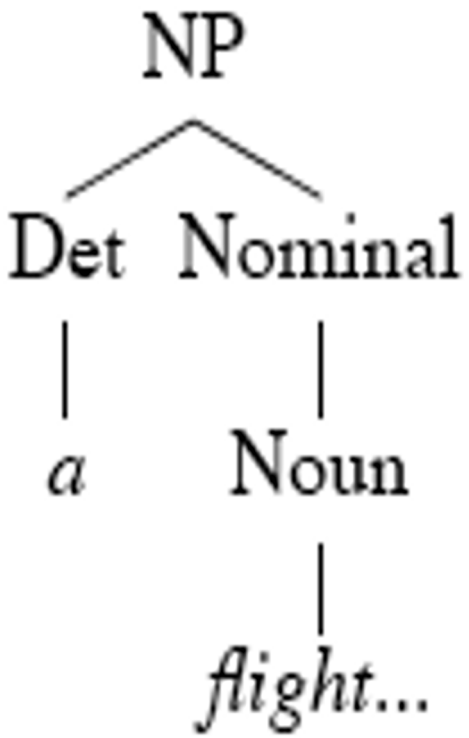
Subtree parsing -> Shared Sub-problems



Shared Sub-problems

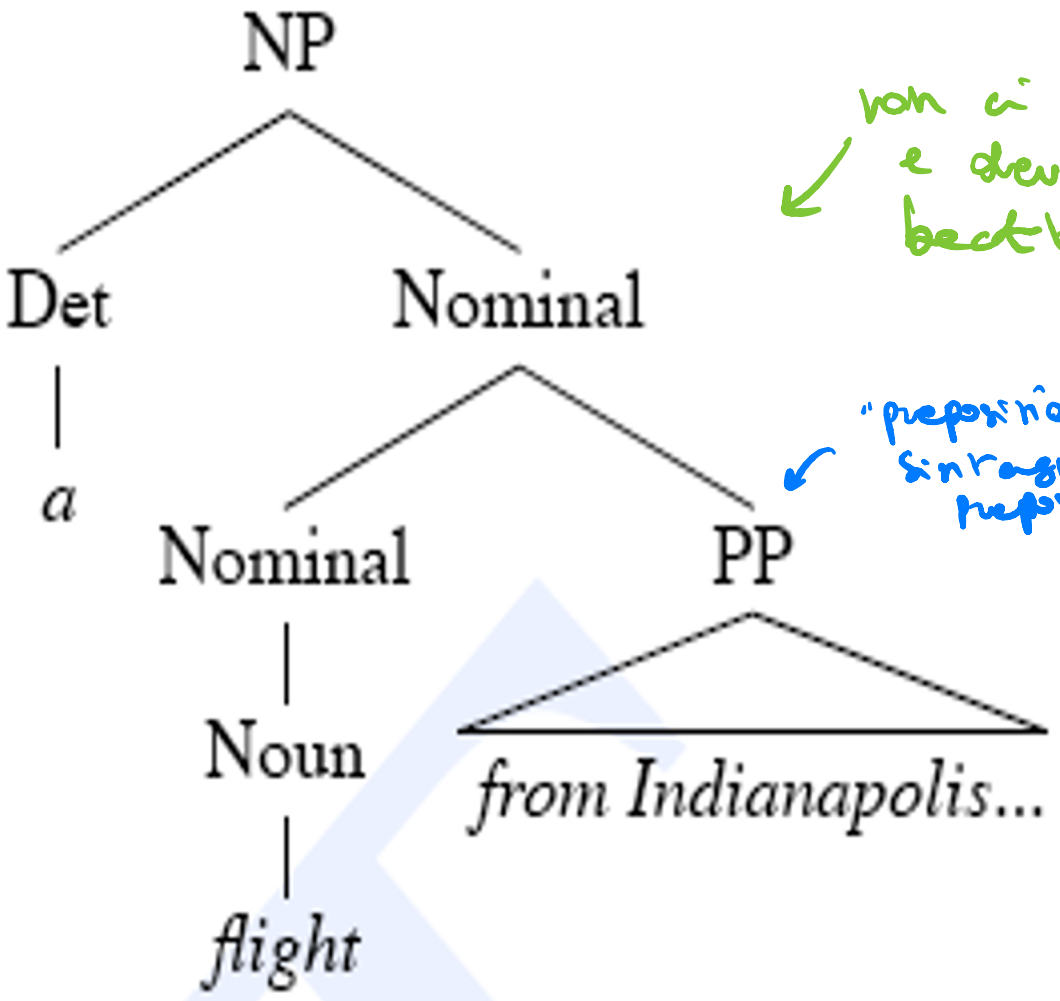
- Top-down parse che ha già espanso la NP
- 2 possibili continuazioni:
  - Nominal -> Noun
  - Nominal -> Nominal PP
- Se scelgo sempre la prima ottengo ...

Shared Sub-problems



poi compone "from Indianapolis" (e' left-to-right)

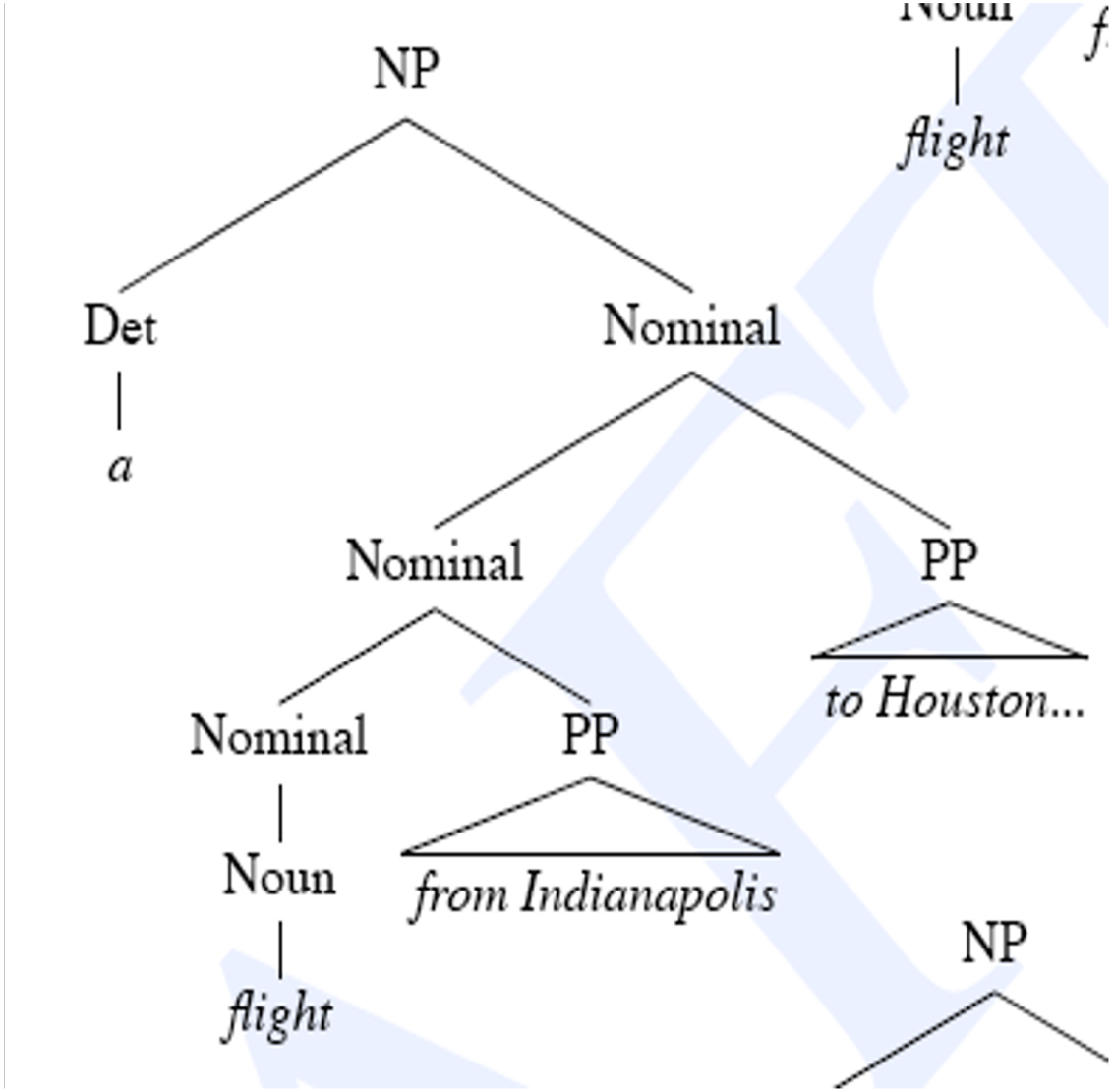
Shared Sub-problems



non ci sta, e devo fare backtracking!

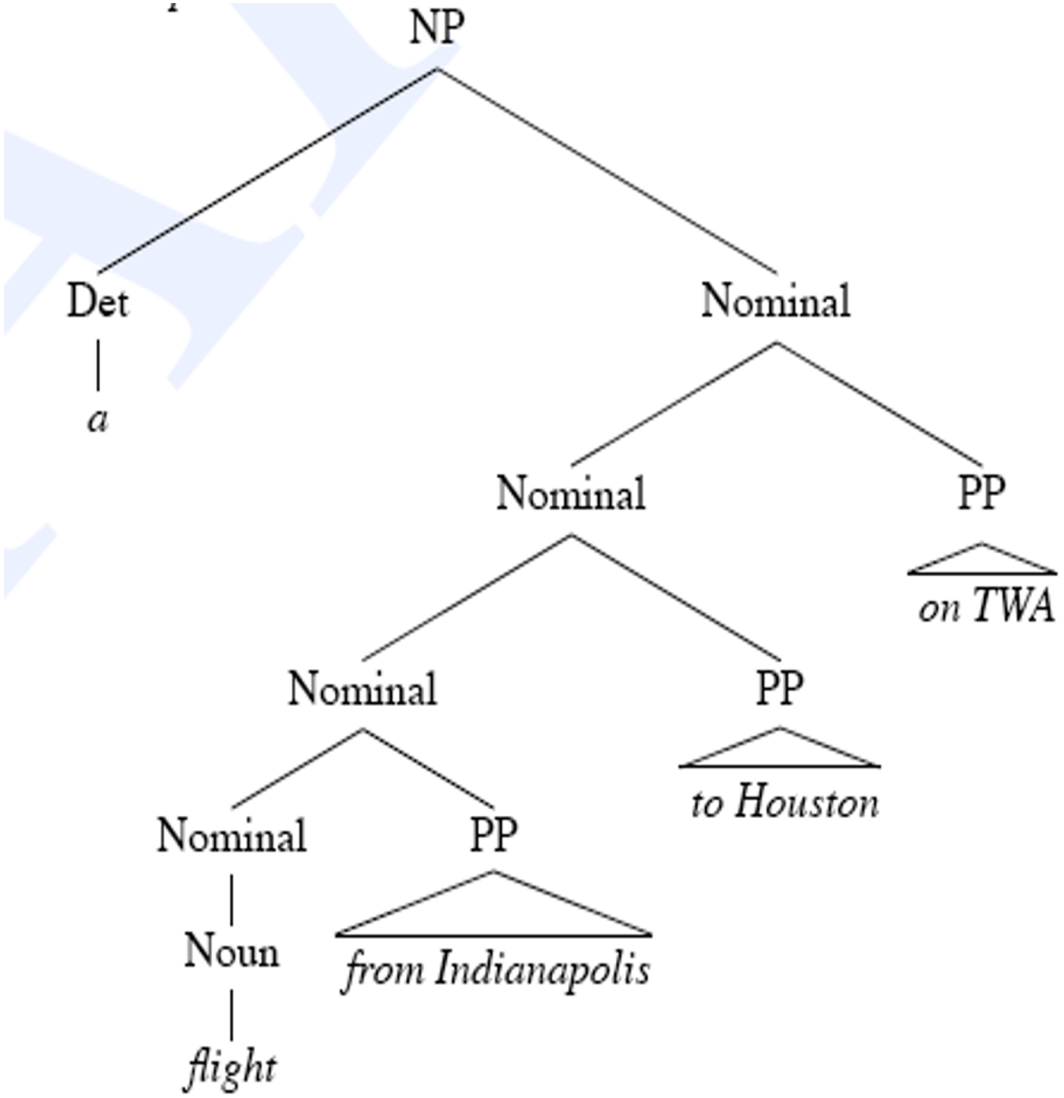
"prepositional phrase", sintagma preposizionale

Shared Sub-problems



stessa cosa...

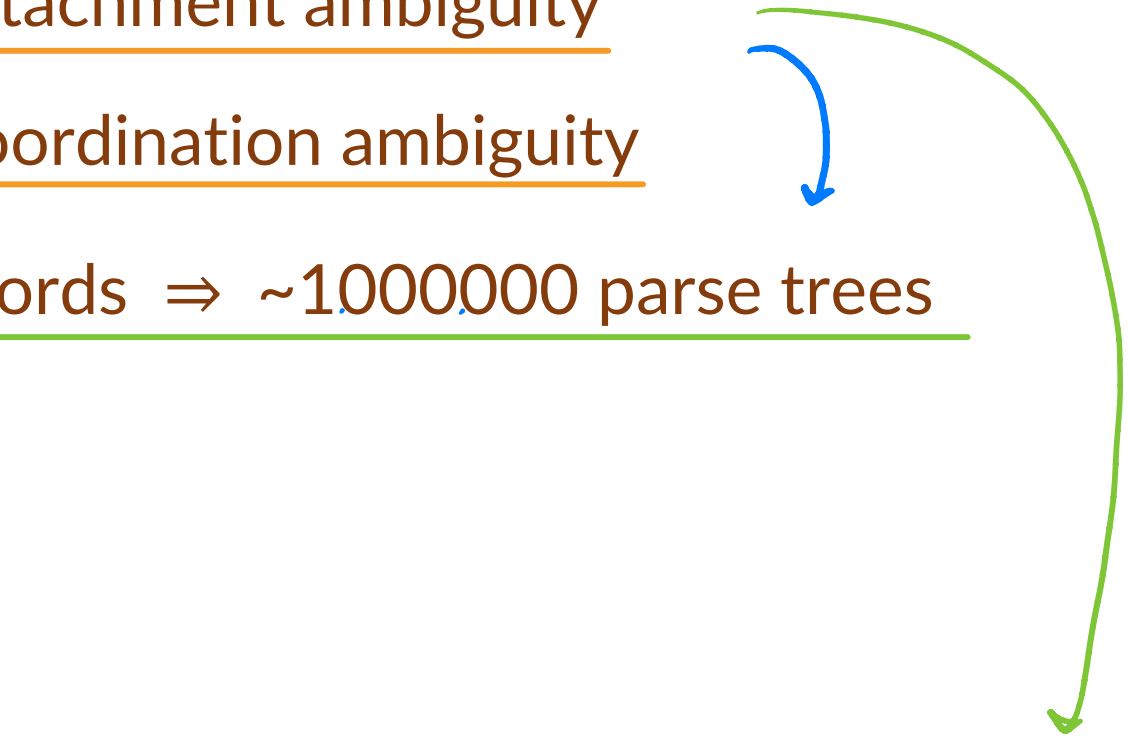
Shared Sub-problems



stessa cosa

Structural ambiguity

- One sentence can have several "legal parse tree"
- Structural (cf. PoS ambiguity)
  1. attachment ambiguity
  2. coordination ambiguity
- 15 words ⇒ ~1000000 parse trees



① Attachment ambiguity (PP)

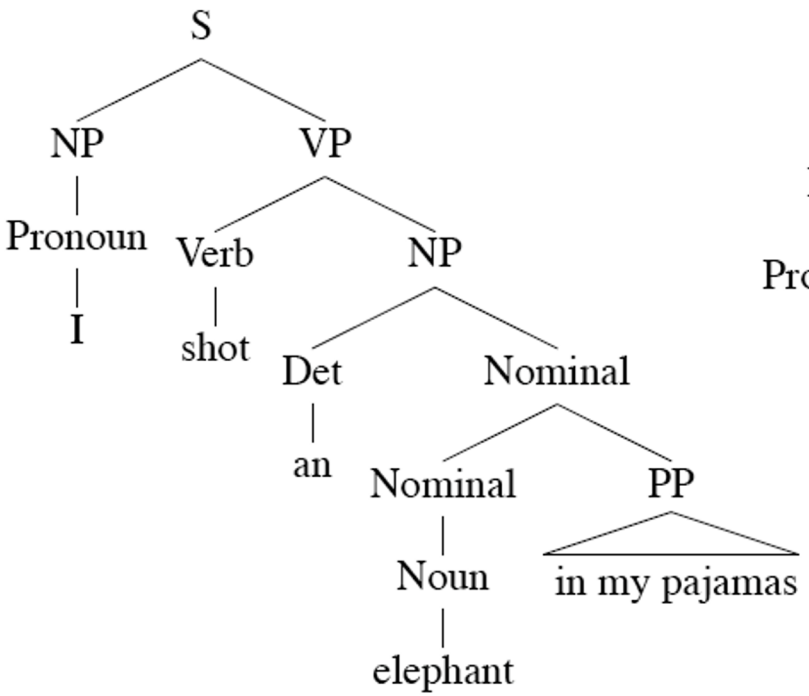
One morning I shot an elephant in my pajamas.

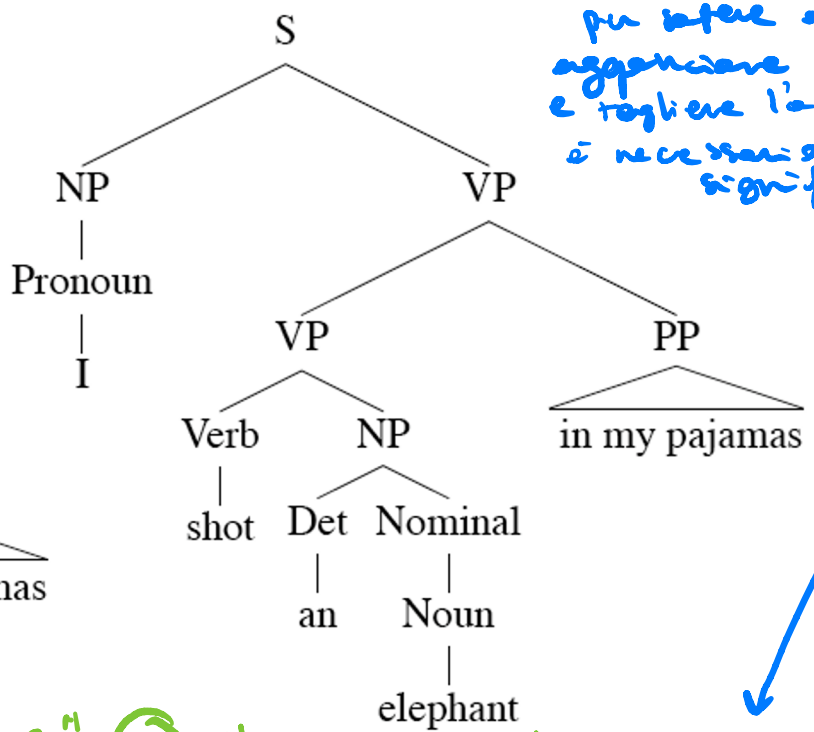
How he got into my pajamas I don't know.

Groucho Marx, *Animal Crackers*, 1930

l'ambiguità sintattica può essere risolta solo dal livello semantico

per capire a cosa agganciare il PP e togliere l'ambiguità è necessario il significato





"ho mangiato la pizza con le acciughe" (vs) "ho mangiato la pizza con la forchetta"

Il livello sintattico e semantico possono avere ambiguità diverse.

Ci sono frasi ambigue a liv. semantico e non a liv. 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 complesso, non c'è una versione giusta

Coordination ambiguity

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!"

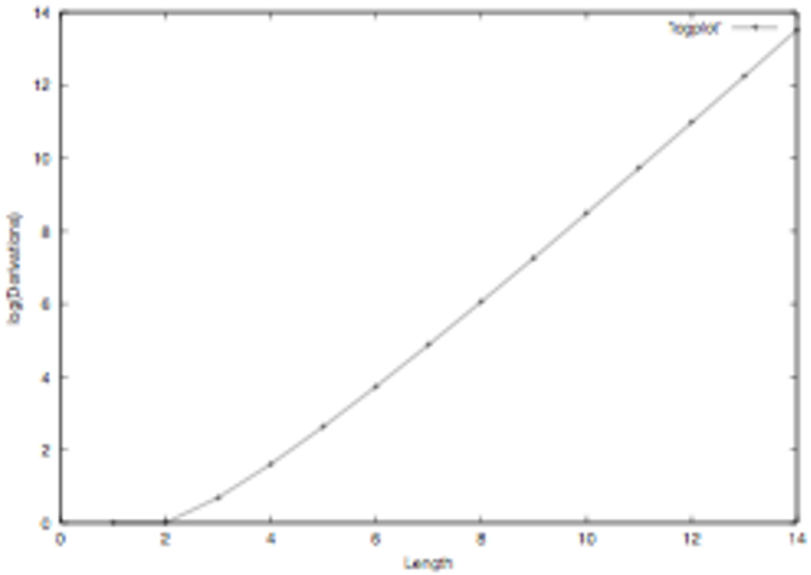
Structural ambiguity

- CFG rules { S → S S , S → a }

grammatica MOLTO ambigua

#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

velocità ..

Catalan numbers (Church and Patil 1982)

- Serie potenza
  - N → natural | language | processing | course
  - N→NN

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

- 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
  - 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 → aSa | bSb | aa | 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)

(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, …

CKY idea

regole non lessicali

premessa: le grammatiche qui sono fatte:

- A → B C : 1 NT sulla sx e 2 NT sulla dx
  oppure
- D → flight : 1 simb. NT sulla sx e 1 simb. terminale sulla dx

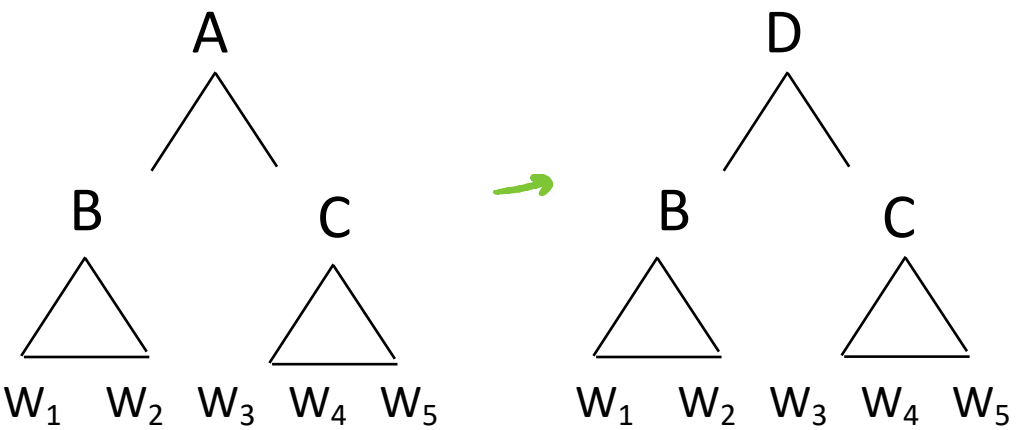
regole lessicali

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

con questo algoritmo

$A \rightarrow B\ C$

$D \rightarrow B\ C$



1

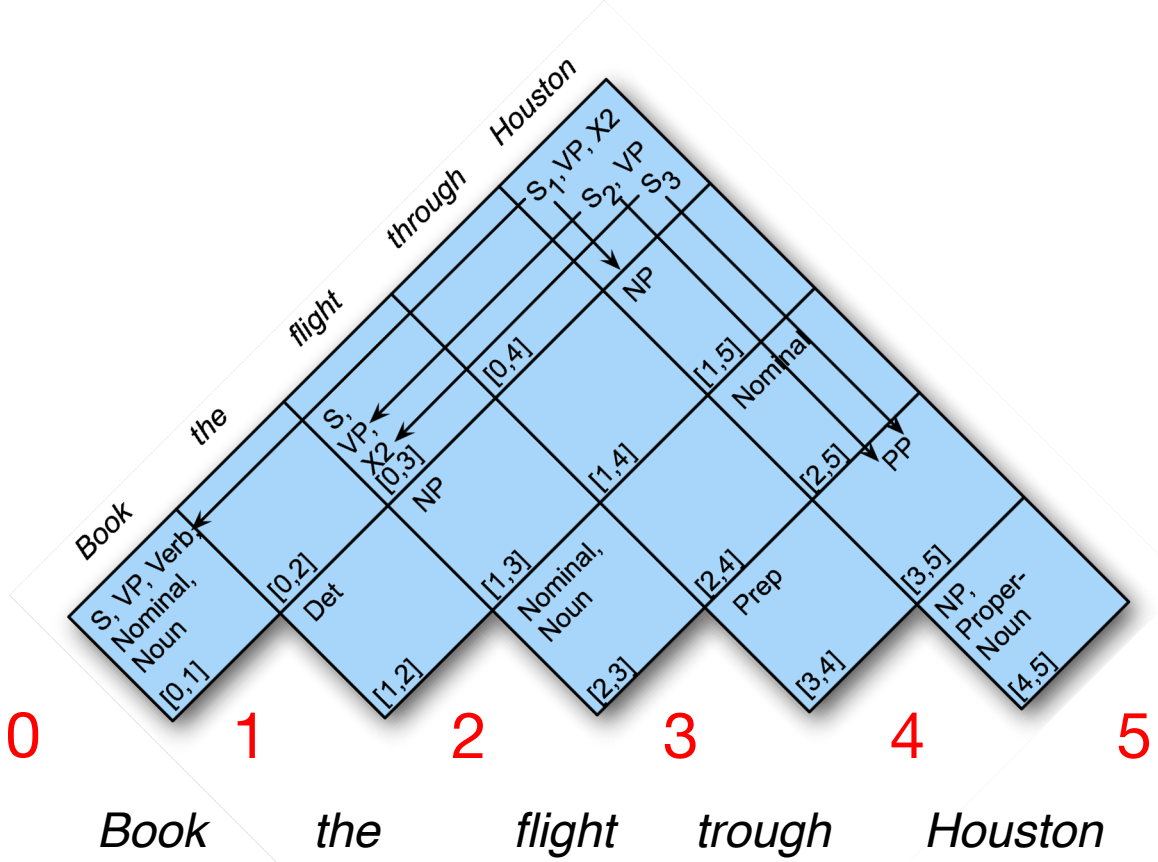
2

se mi accorgo che non è la strada giusta, invece che fare backtracking e buttare tutto, mantengo i sotto alberi comuni B C

CF in Forma Normale di Chomsky

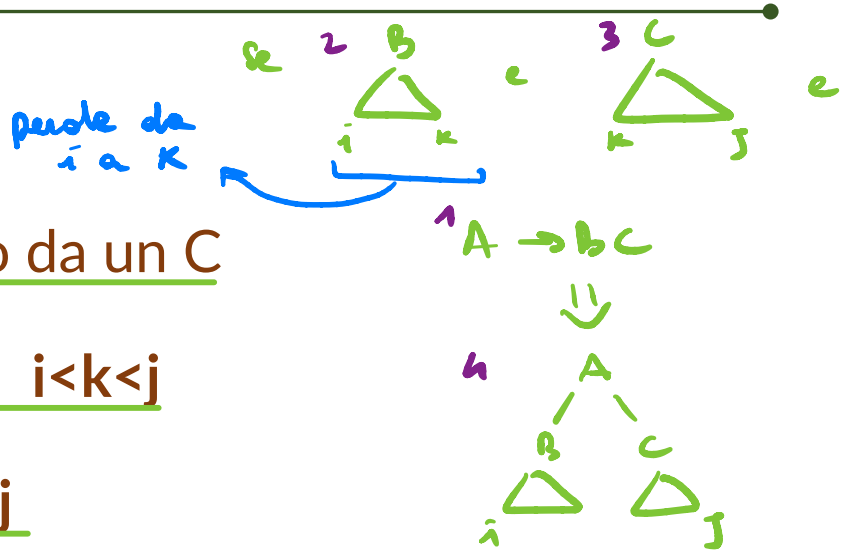
- Copy all conforming rules to the new grammar unchanged,
- Convert terminals within rules to dummy non-terminals,
- Convert unit-productions,
- Binarize all rules and add to new grammar.
- Epsilon free

Idea della table



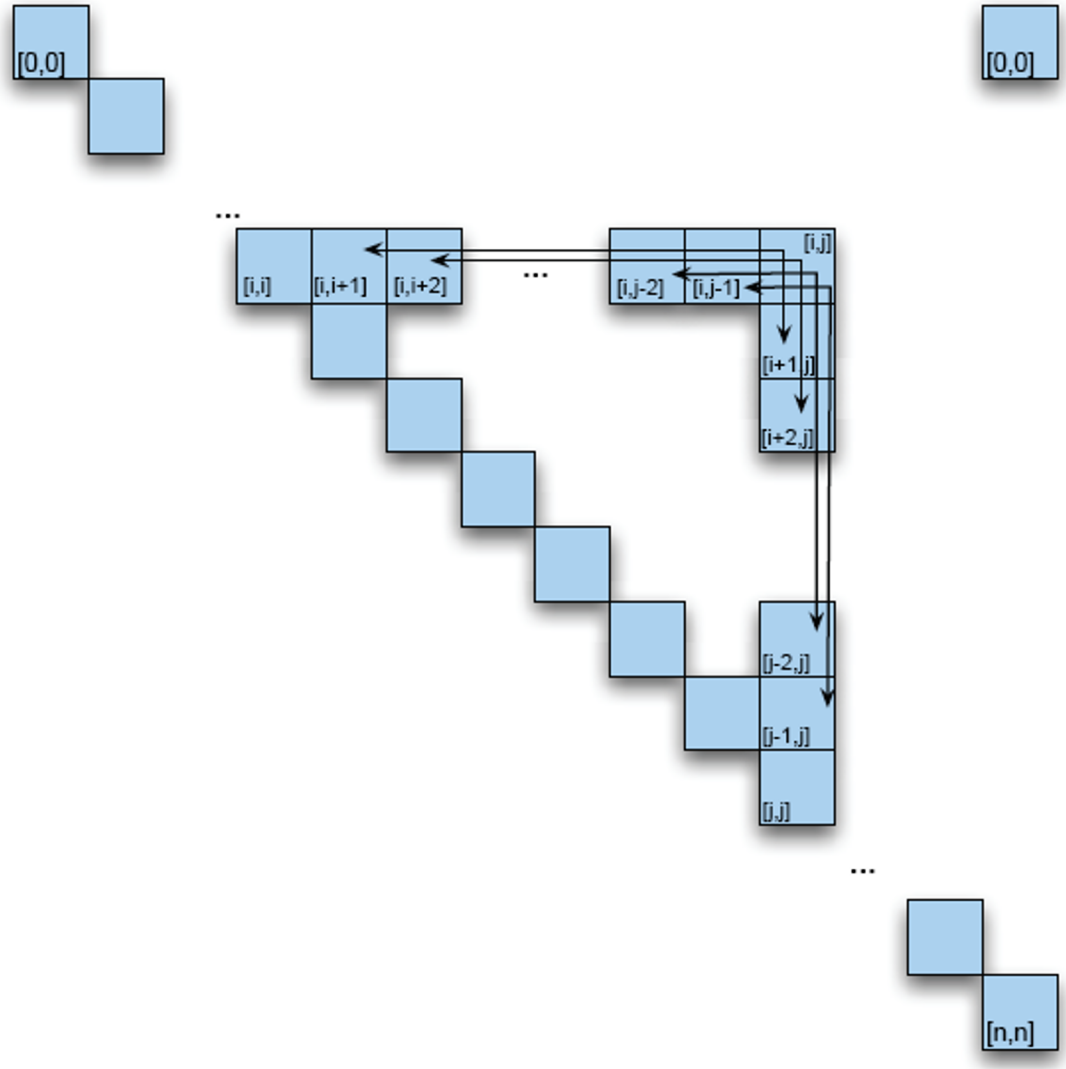
CKY

$_0Book_1that_2flight_3$



- 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 -> **S in [0,n]**

Filling the table



Grammatica L1

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

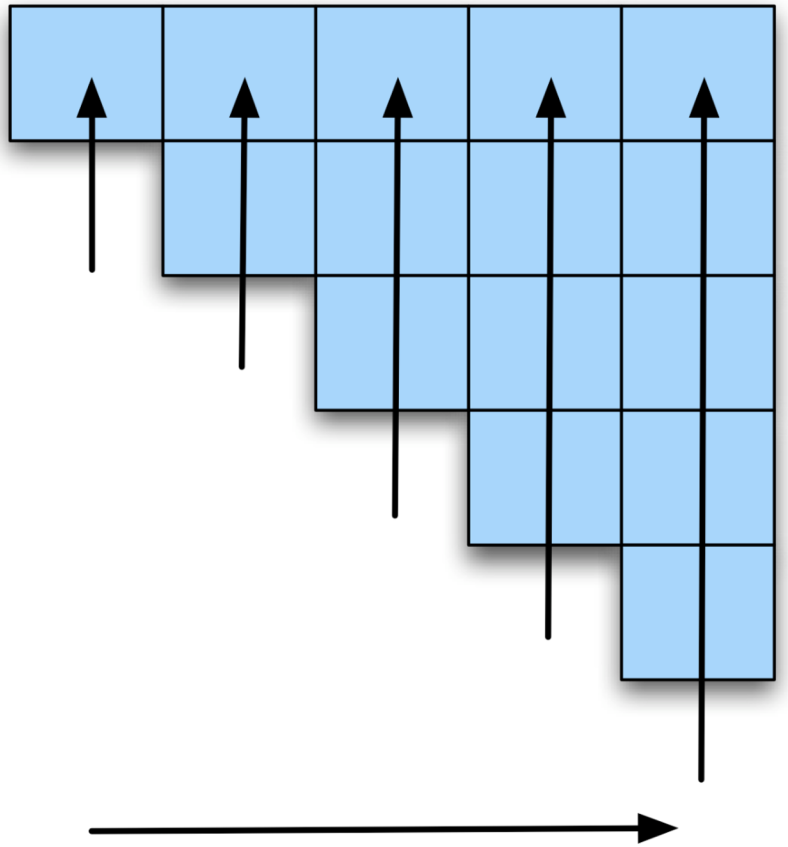
Grammatica L1 e normal form

$S \rightarrow NP\ VP$	$S \rightarrow NP\ VP$
$S \rightarrow Aux\ NP\ VP$	$S \rightarrow X1\ VP$
	$X1 \rightarrow Aux\ NP$
$S \rightarrow VP$	$S \rightarrow book \mid include \mid prefer$
	$S \rightarrow Verb\ NP$
	$S \rightarrow X2\ PP$
	$S \rightarrow Verb\ PP$
	$S \rightarrow VP\ PP$
$NP \rightarrow Pronoun$	$NP \rightarrow I \mid she \mid me$
$NP \rightarrow Proper\text{-}Noun$	$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$	$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow Noun$	$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal\ Noun$	$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$	$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow Verb$	$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$	$VP \rightarrow Verb\ NP$
$VP \rightarrow Verb\ NP\ PP$	$VP \rightarrow X2\ PP$
	$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$	$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$	$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$	$PP \rightarrow Preposition\ NP$

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*

inizializz. diagonale (cfr. Viterbi) →

for $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do**
  $table[j-1, j] \leftarrow \{A \mid A \rightarrow words[j] \in grammar\}$
  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 $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do**

  $table[j-1, j] \leftarrow \{A \mid A \rightarrow words[j] \in$

  for $i \leftarrow$ **from** $j-2$ **downto** 0 **do**

    for $k \leftarrow i+1$ **to** $j-1$ **do**

      $table[i,j] \leftarrow table[i,j] \cup$

$\{A \mid A \rightarrow BC \in grammar,$

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

$C \in table[k,j]\}$

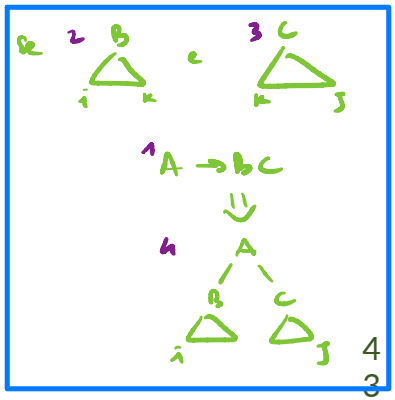
Looping over the columns

Filling the bottom cell

Filling row i in column j

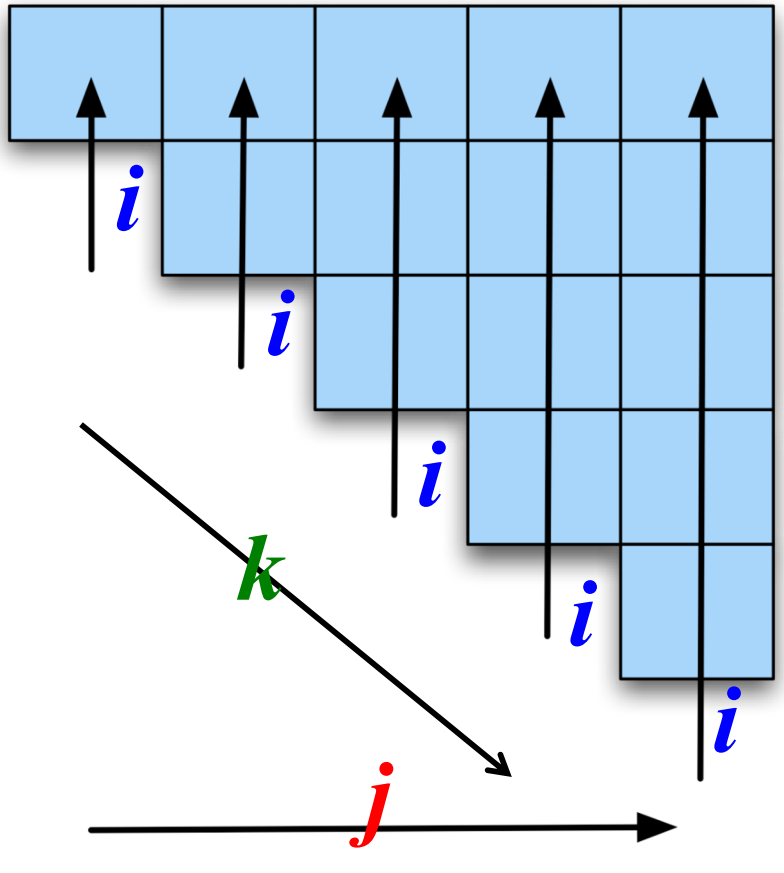
Looping over the possible split locations between i and j.

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].



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

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



$[\, j : 1\ldots n \; [\, i : j\text{-}2\ldots 0 \; [\, k : i{+}1\ldots j{-}1 \,]\,]\,]$

#parole

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

PP → Preposition NP

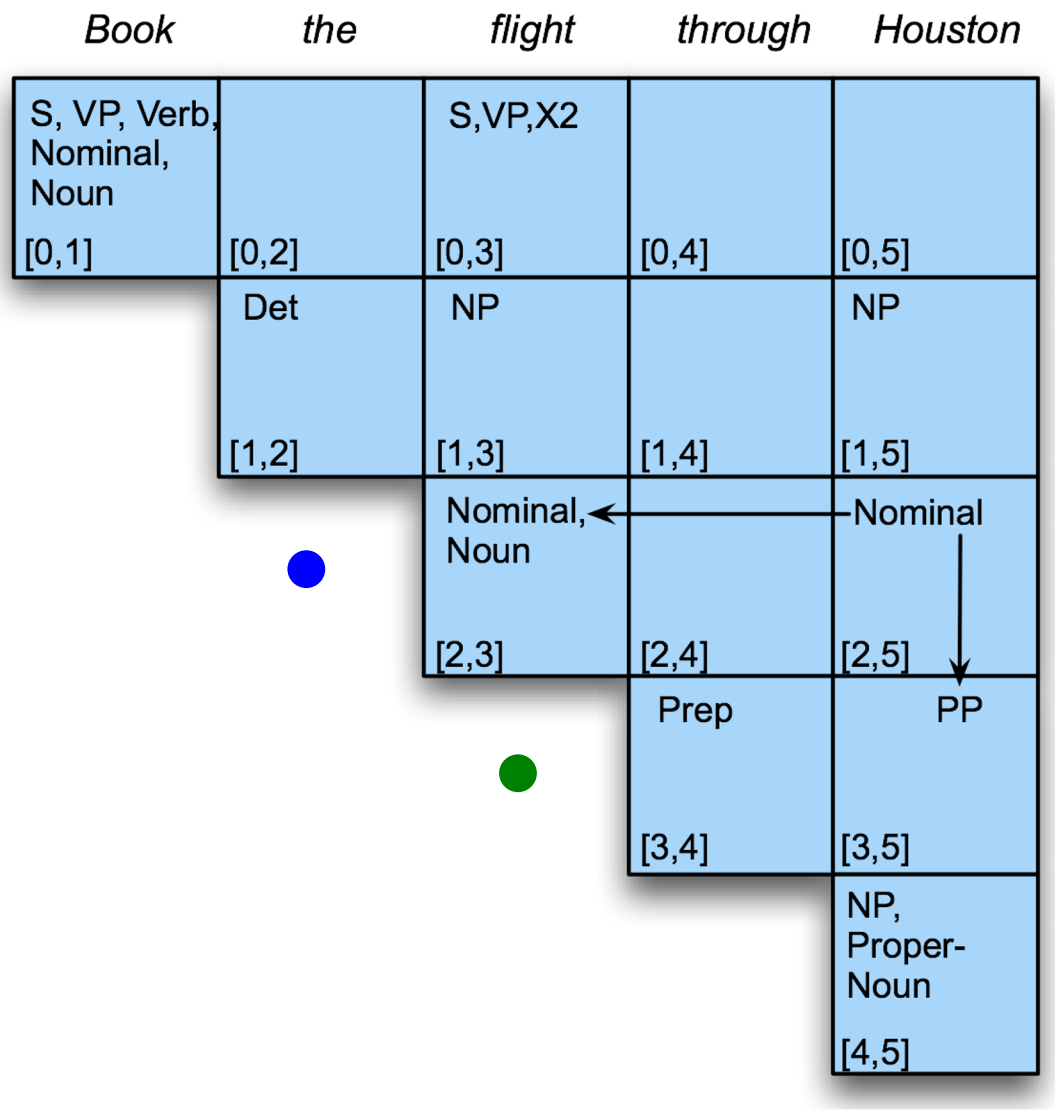
j=5

i=3

k=4

Consideriamo solo la colonna 5

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

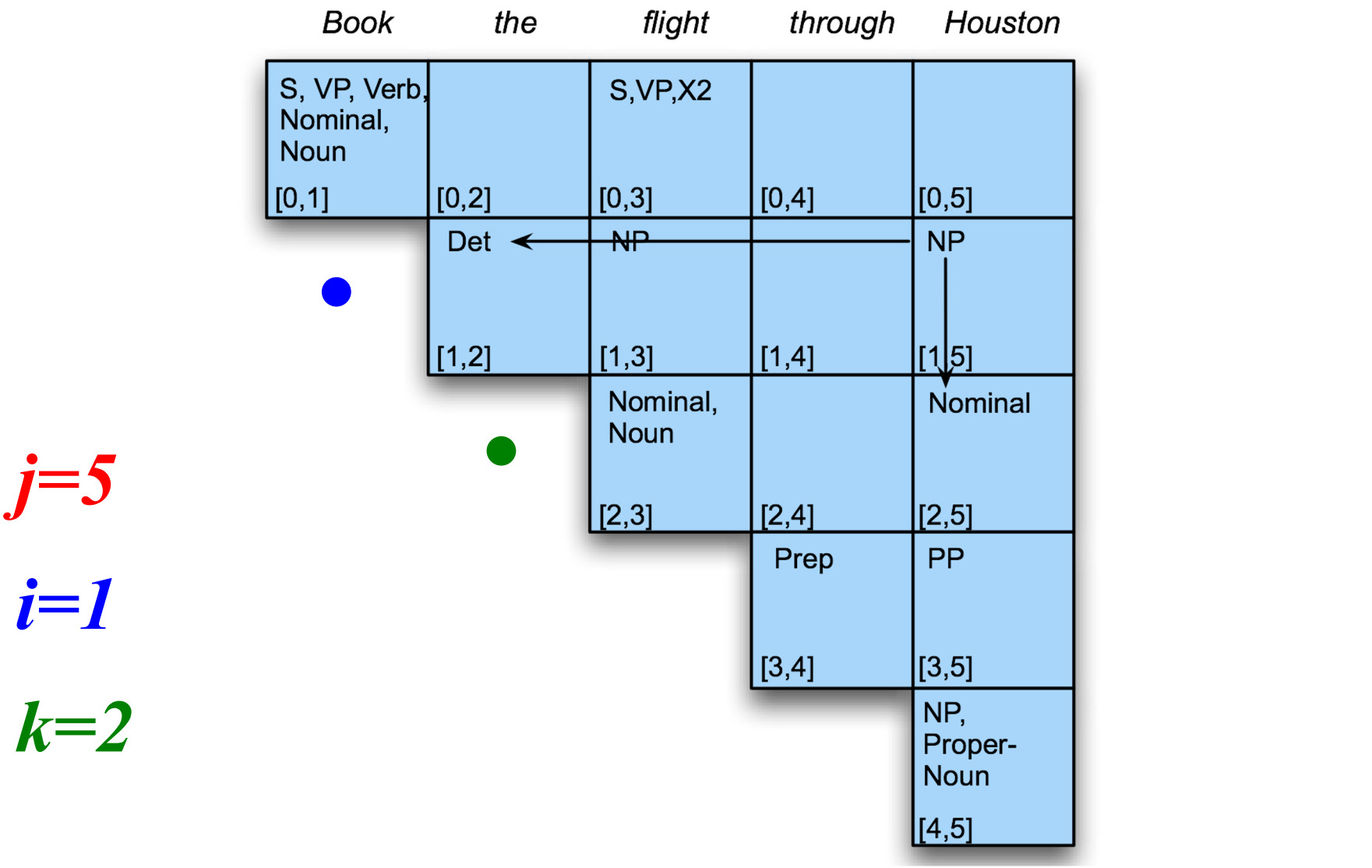


j=5

i=2

k=3

e [*j* : 1...n [*i* : j-2...0 [*k* : i+1...j-1]]]



j=5

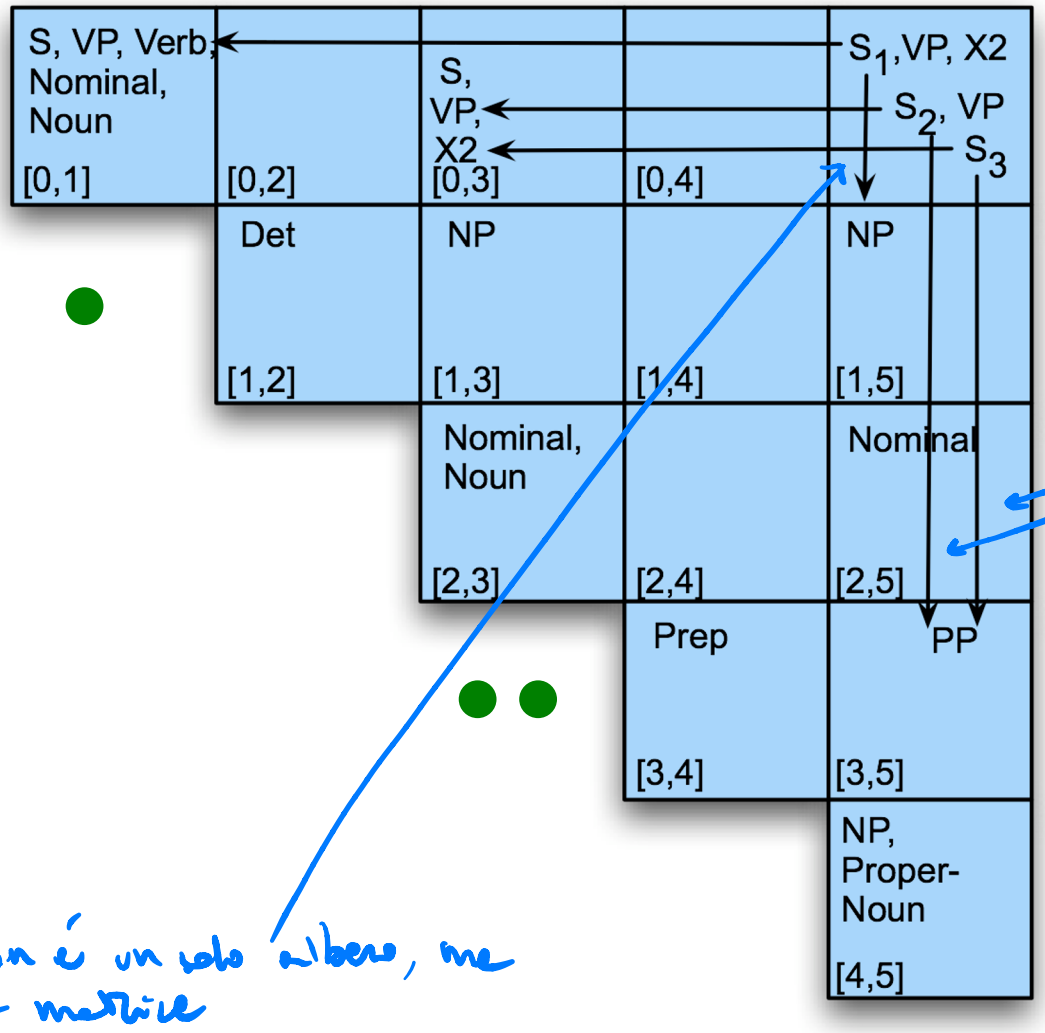
i=1

k=2

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

Riconoscitore: dice se la frase appartiene o meno alla grammatica

è sì se alla fine dà per uno in [0,S] un "S"



vs

Parser: fa qualcosa in più, dice qual è l'albero giusto/gli alberi giusti (le frasi potrebbero essere ambigue)

per questo usa dei back pointer

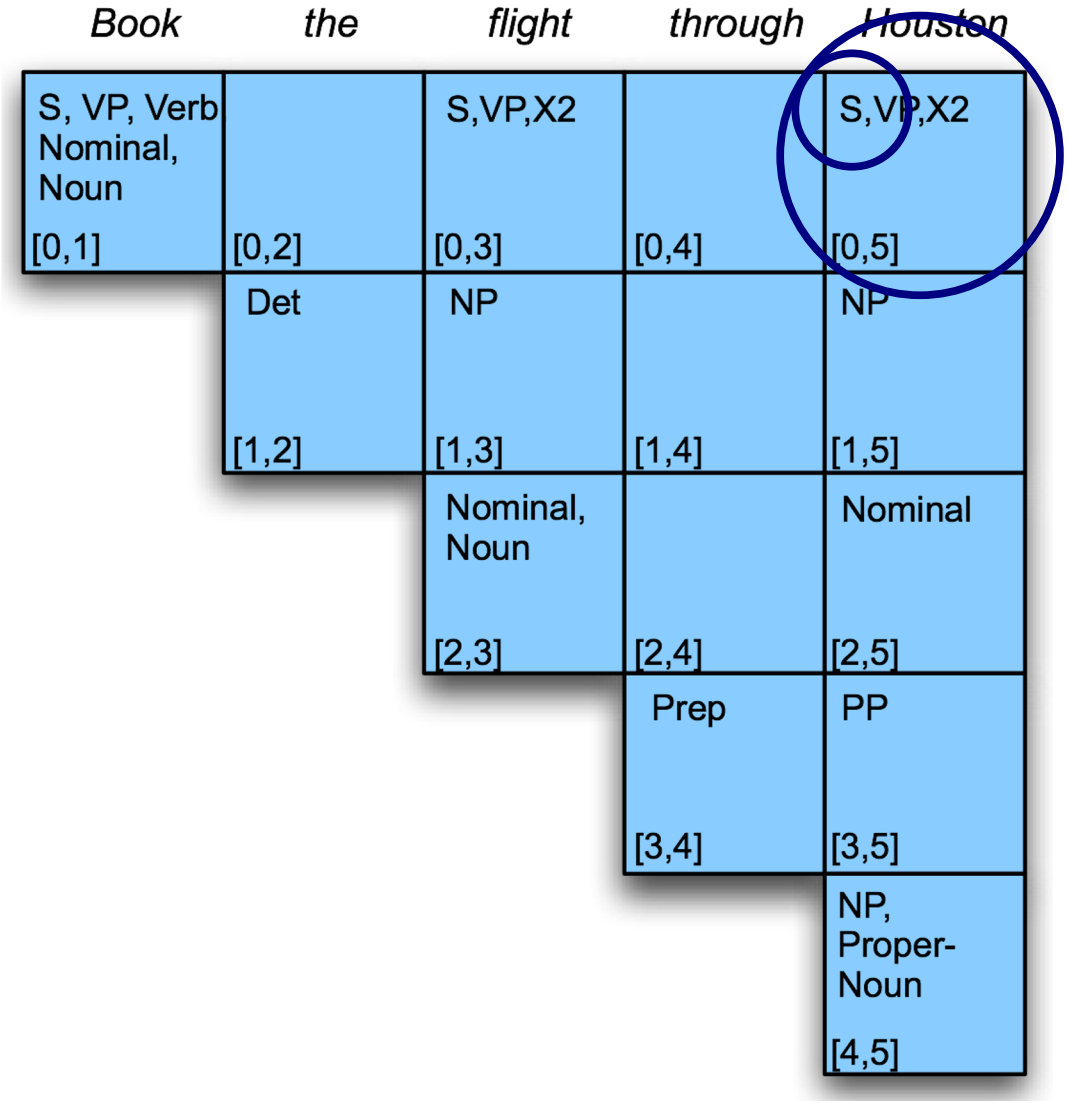
$j=5$

$i=0$

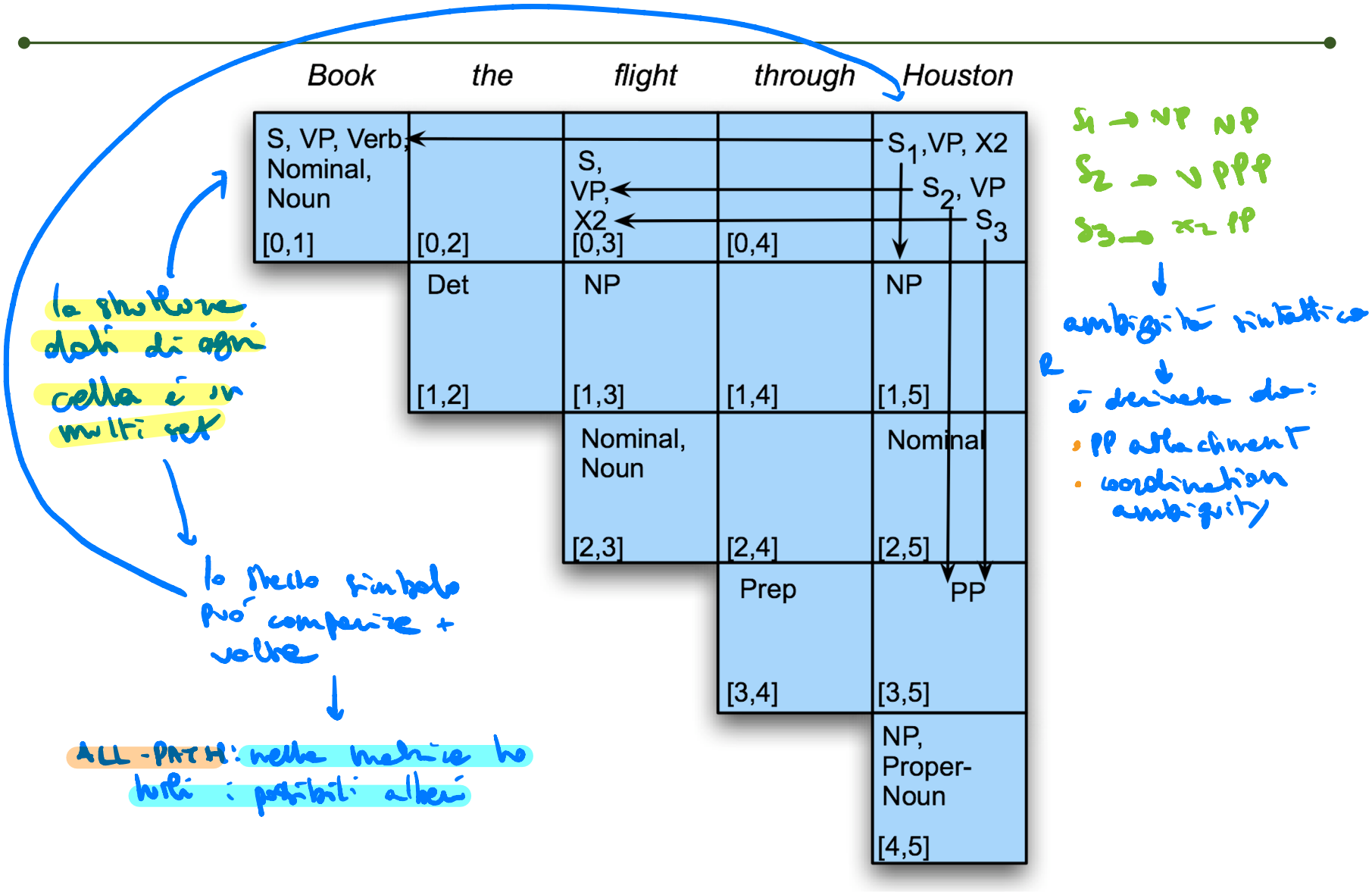
$k=1,3,3$ (2)

l'output non è un solo albero, ma tutta la matrice

Successo!


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]

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

$_0Paolo_1ama_2Francesca_3dolcemente_4$

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

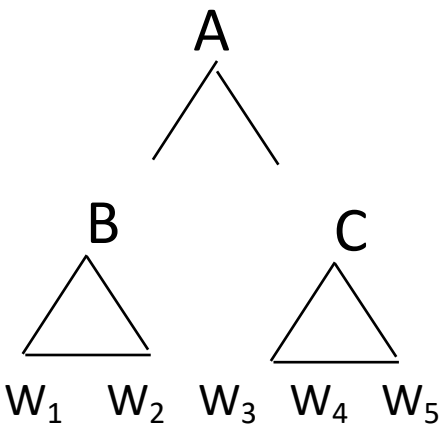
associa una probab. a ogni regola di riscrittura

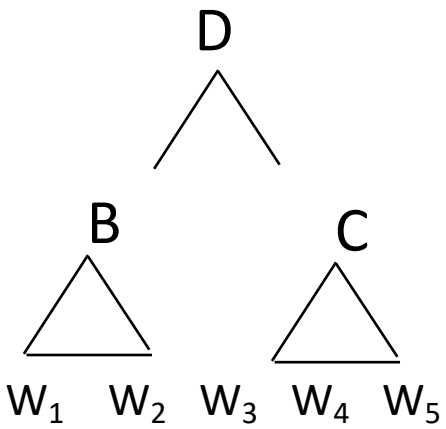
$A \rightarrow BC \quad [p_A]$

$D \rightarrow BC \quad [p_D]$

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

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





Probabilistic CFG

G=(Σ,V,S,P)

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

la somma di tutte le prob di $A \rightarrow \beta$ è 1

è una distrib. di probabilità

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

PCFG

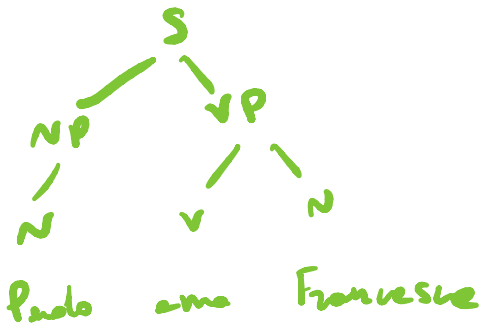
$S \rightarrow NP\ VP$	[.80]	$Det \rightarrow that\,[.05] \mid the\,[.80] \mid a\,[.15]$	
$S \rightarrow Aux\ NP\ VP$	[.15]	$Noun \rightarrow book$	[.10]
$S \rightarrow VP$	[.05]	$Noun \rightarrow flights$	[.50]
$NP \rightarrow Det\ Nom$	[.20]	$Noun \rightarrow meal$	[.40]
$NP \rightarrow Proper\text{-}Noun$	[.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]
$Nom \rightarrow Proper\text{-}Noun\ Nom$	[.05]	$Aux \rightarrow do$	[.30]
$VP \rightarrow Verb$	[.55]	$Proper\text{-}Noun \rightarrow TWA$	[.40]
$VP \rightarrow Verb\ NP$	[.40]	$Proper\text{-}Noun \rightarrow Denver$	[.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 regola

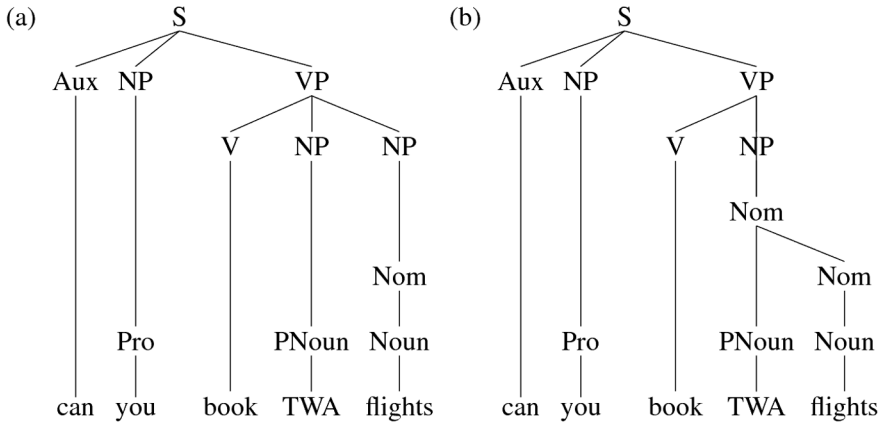
$$P(T,S) = \prod_{node \in T} P(rule(n))$$



dobbiamo recuperare le regole che generano l'albero

le ricaviamo livello - livello (perché è una CF)

PCFG



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

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

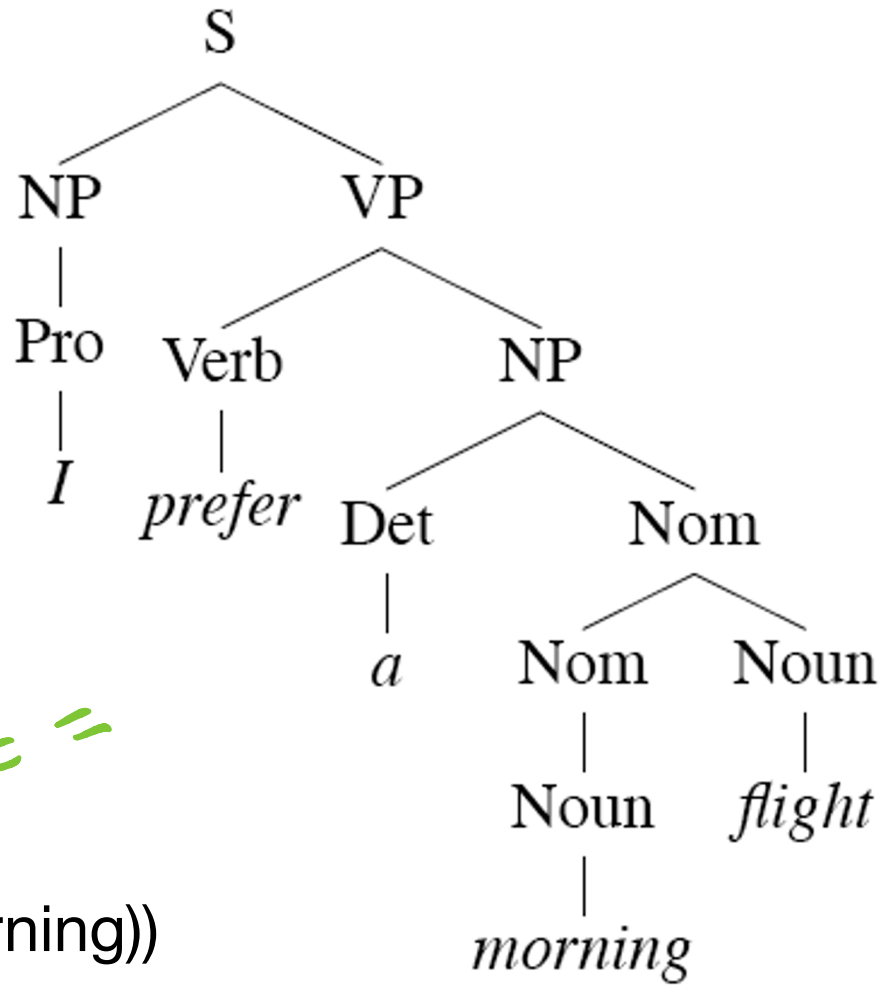
$P(T_a) = .15 * .4 * .05 * .05 * .35 * .75 * .4 * .4 * .4 * .3 * .4 * .5 = 1.5 \times 10^{-6}$

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

Da dove prendo le probabilità?

- Treebanks -> banche di alberi
  - 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



(S (NP (Pro I))

(VP (Verb prefer)

(NP (Det a)

(Nom (Nom (Noun morning))

(Noun flight)))))

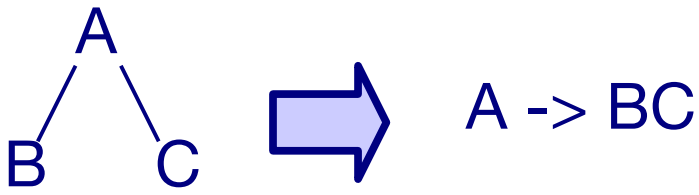
Penn TB

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**:



- 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:

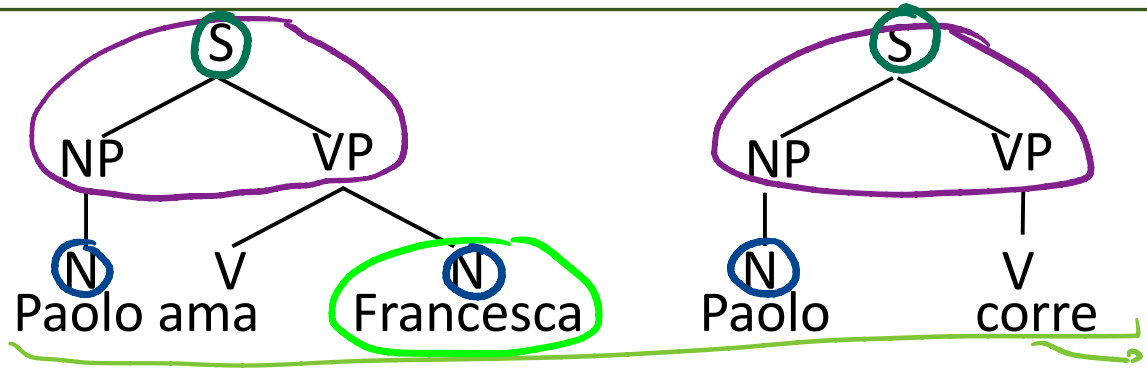
$$P(\alpha \rightarrow \beta|\alpha) = \frac{\text{Count}(\alpha \rightarrow \beta)}{\sum_\gamma \text{Count}(\alpha \rightarrow \gamma)} = \frac{\text{Count}(\alpha \rightarrow \beta)}{\text{Count}(\alpha)}$$

di apparizioni di una regola nel TB

di apparizioni della testa nel TB

assomiglia all'HMM

Treebank Grammars (PCFG)



P(A→β)=Count(A→β)/Count(A)

P(S→NP VP) =2/2=1 P(NP→N) =2/2=1

P(VP→V N) =1/2=.5 P(VP→V) =1/2=.5

P(N→Paolo) =2/3=.66 P(N→Francesca) =1/3=.33

P(V→corre) =1/2=.5 P(V→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($back[1, \text{LENGTH}(words), S]$), $table[1, \text{LENGTH}(words), S]$

è pseudo codice: oltre al numero (la prob. + alta) mantengo anche il back pointer al sotto-albero

se il nuovo è + prob. lo sostituisco al vecchio

vs CKY senza prob. dove c'era una Union

è un oracolo probabilistico

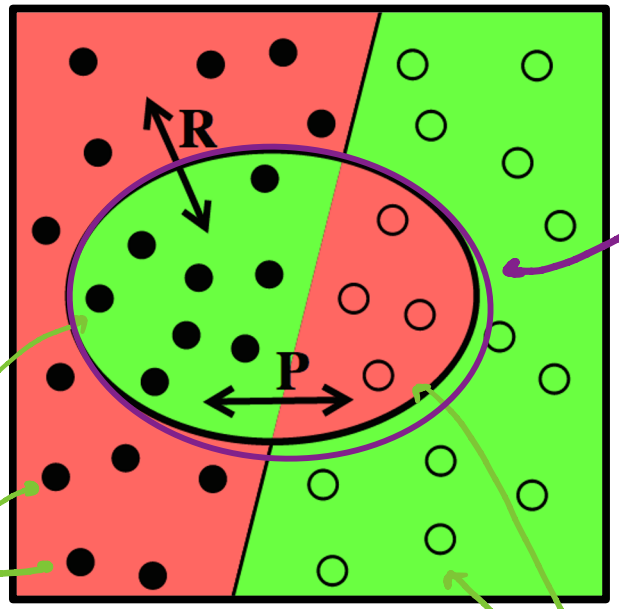
pro: - rimuove un po' di ambiguità → espl. comb.
con: - è un po' + lento

è un beam-search

nella tabella ora metto delle distrib. di prob.

Valutazione

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



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

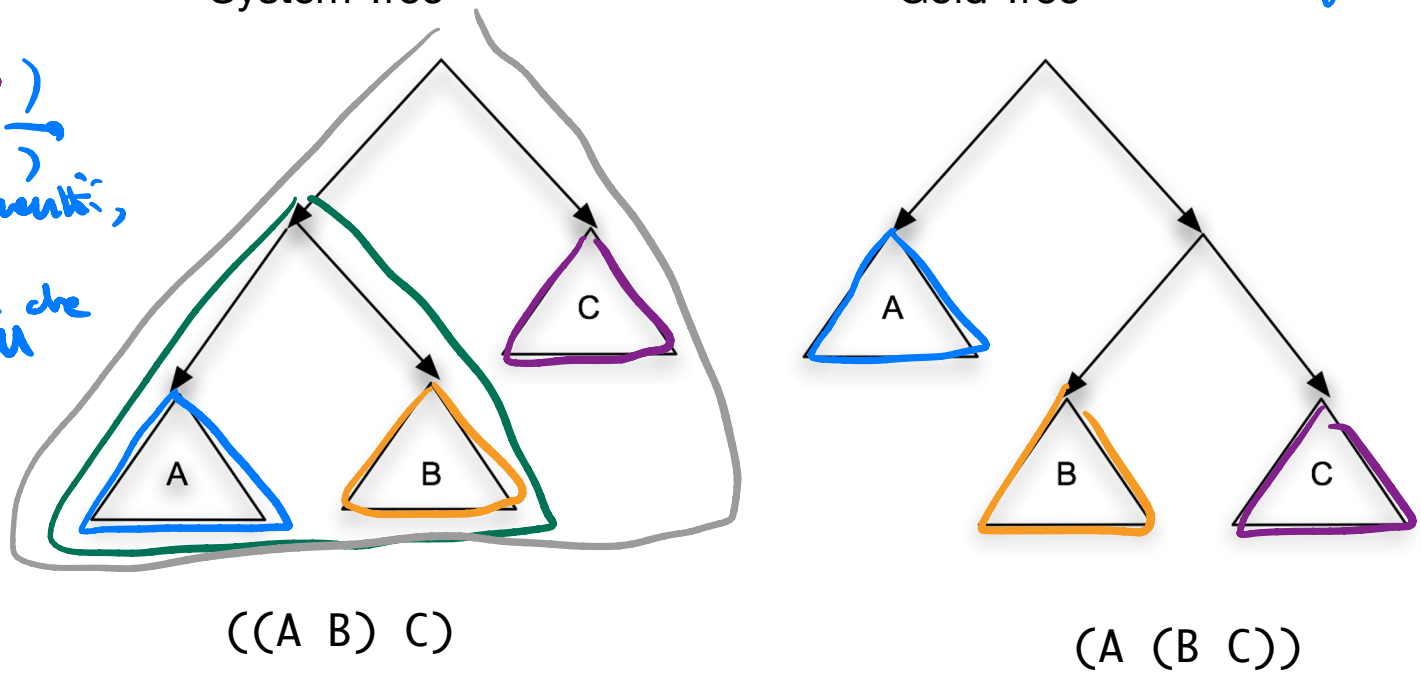
$$R = \frac{T^{\bullet}}{T^{\bullet} + F^{\circ}}$$

Valutazione per le PCFG

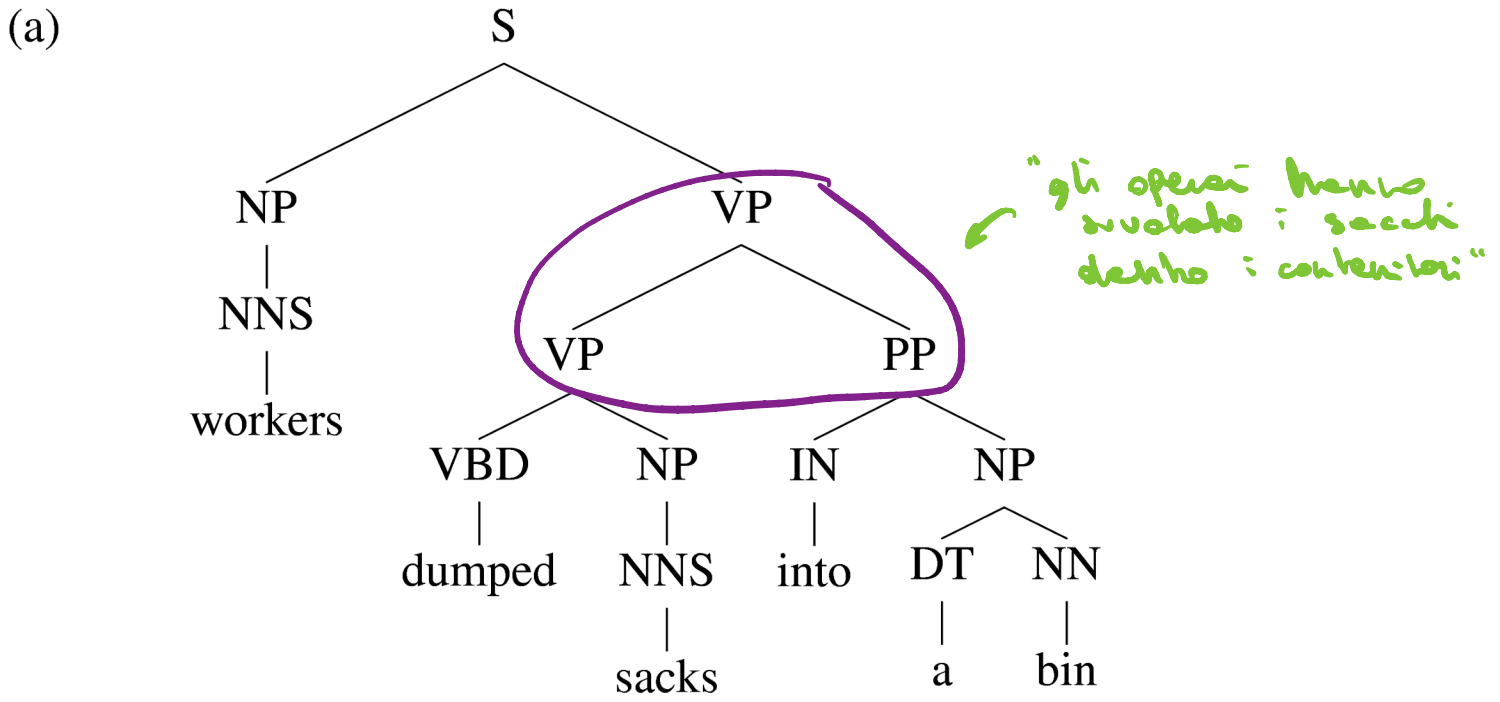
- Precision
  - Quale percentuale di subtree del system tree sono anche nel gold tree? >
  - *Quanto di quello prodotto è giusto?*
- Recall
  - Quale percentuale di subtree del golden tree sono anche nelsystem tree? ->
  - *Quanto di quello che avremmo dvuto produrre abbiamo prodotto?*

Parseval

- Crossing brackets



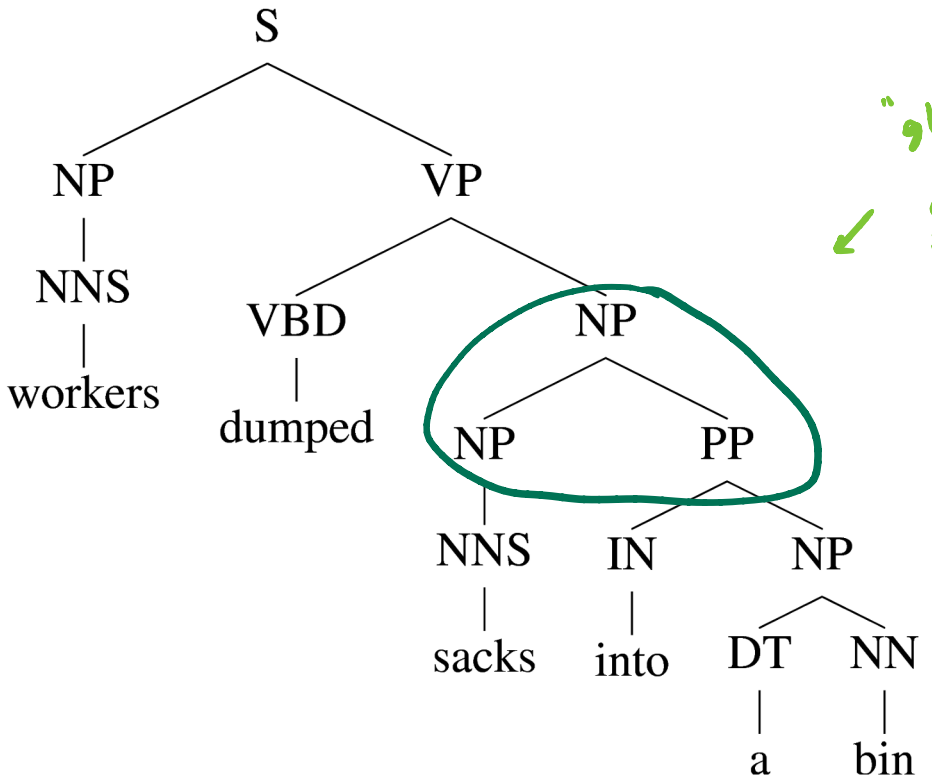
Problem with PCFG



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

Problem with PCFG

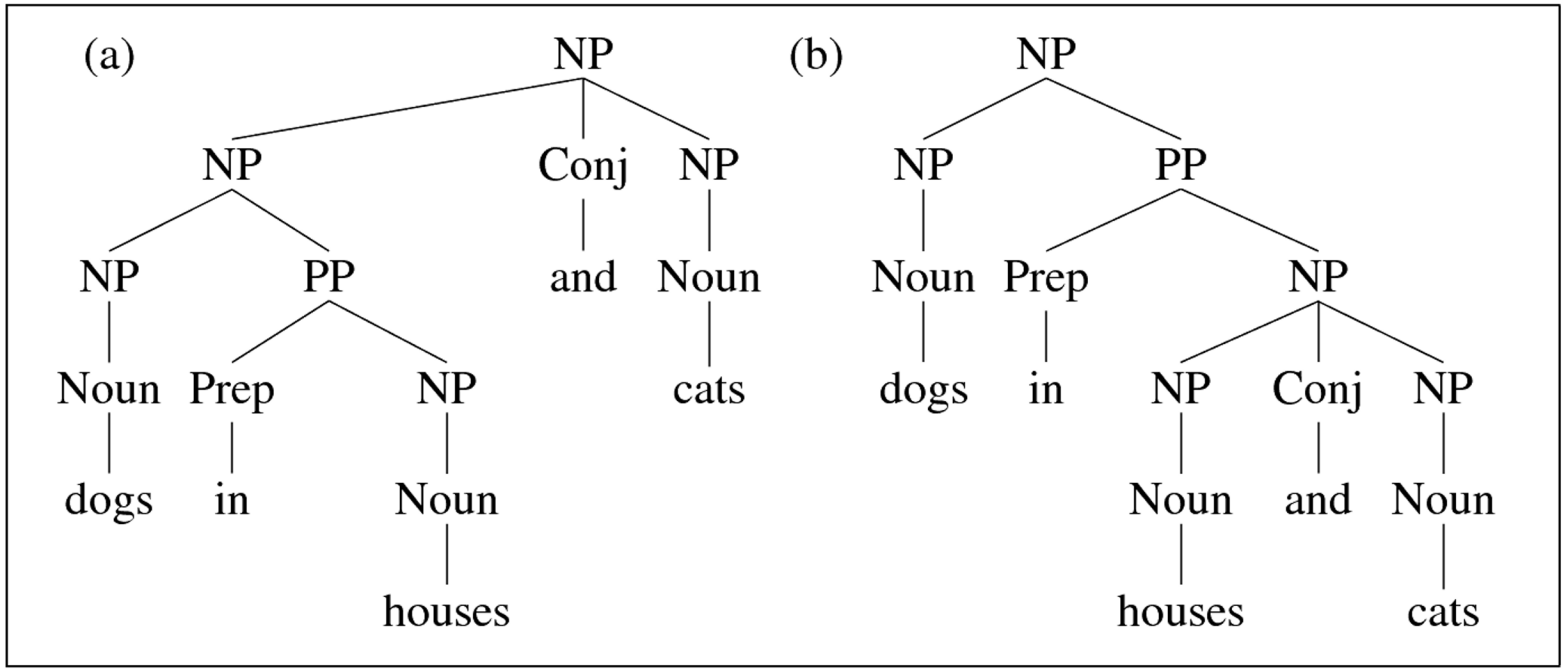
(b)



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

Problem with PCFG

anche questi 2 alberi, che hanno una semantica molto diversa, hanno la stessa prob. dal punto di vista sintattico (mentre vorremmo che fosse diversa)



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

Lexicalized PCFG

Ogni regola CF è potenziata con informazione circa le **heads** dei costituenti coinvolti

$A \rightarrow BC$ $\quad\quad$ $A(head_A) \rightarrow B(head_B)\ C(head_C)$

Punto di unione tra formalismi sintattici a costituenti e a dipendenza

Lexicalized PCFG

uso regole linguistiche x associare a ogni nodo un simb. NT, ma anche quel è il termin. nella frase del TB che ha + influito nella costruzione di quel NT

- VP → VBD NP PP

VP(dumped) → VBD(dumped) NP(sacks) PP(into) [3x10^{-10}]

VP(dumped) → VBD(dumped) NP(cats) PP(into) [8x10^{-11}]

VP(dumped) → VBD(dumped) NP(hats) PP(into) [4x10^{-10}]

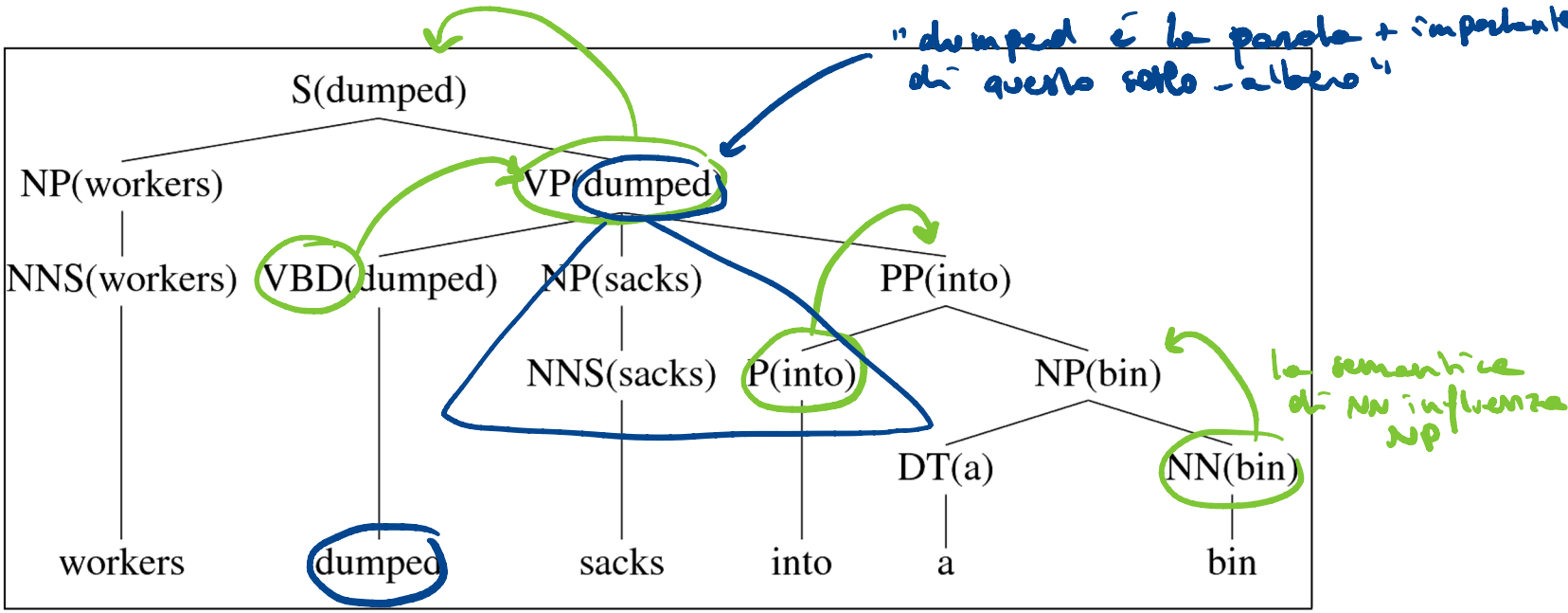
VP(dumped)→VBD(dumped) NP(sacks) PP(above) [1x10^{-12}]

delle stesse regole ora ne ho 4

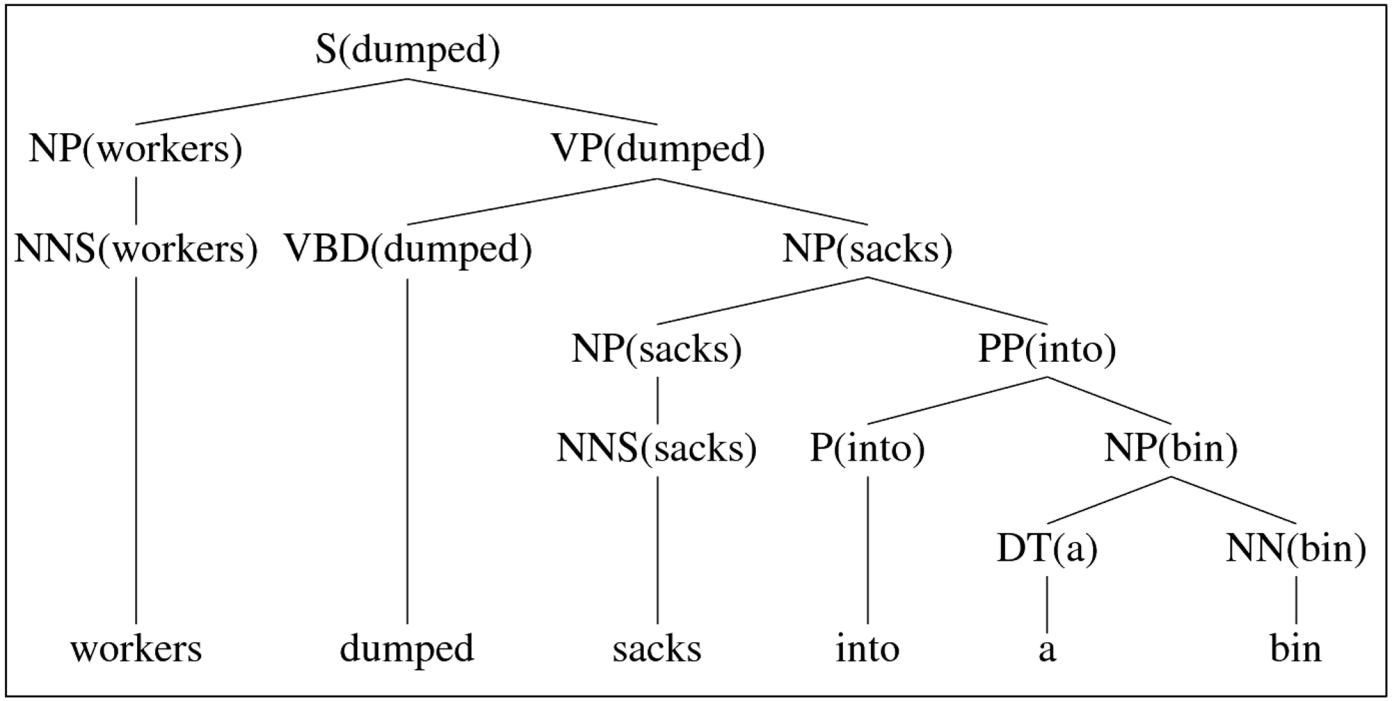
PERCOLAZIONE: faccio emergere l'elemento lessicale + importante al genitore

con il numero di regole esplode => devo usare una tecnica di smoothing molto raffinata

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

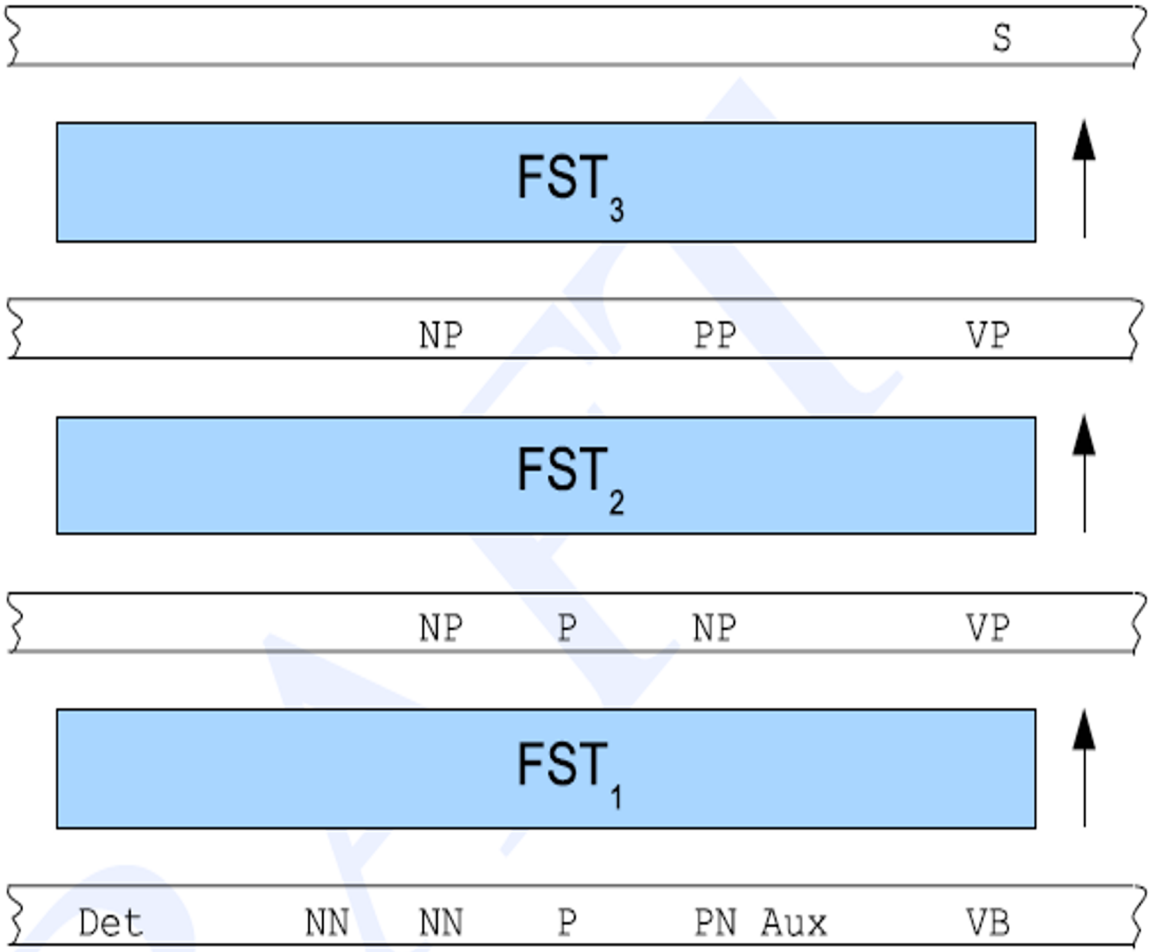
$$NP \rightarrow (Det)\ Noun^*\ Noun$$

$$NP \rightarrow Proper\text{-}Noun$$

$$VP \rightarrow Verb$$

$$VP \rightarrow Aux\ Verb$$

Cascata di trasducers: approssimo una CFG



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

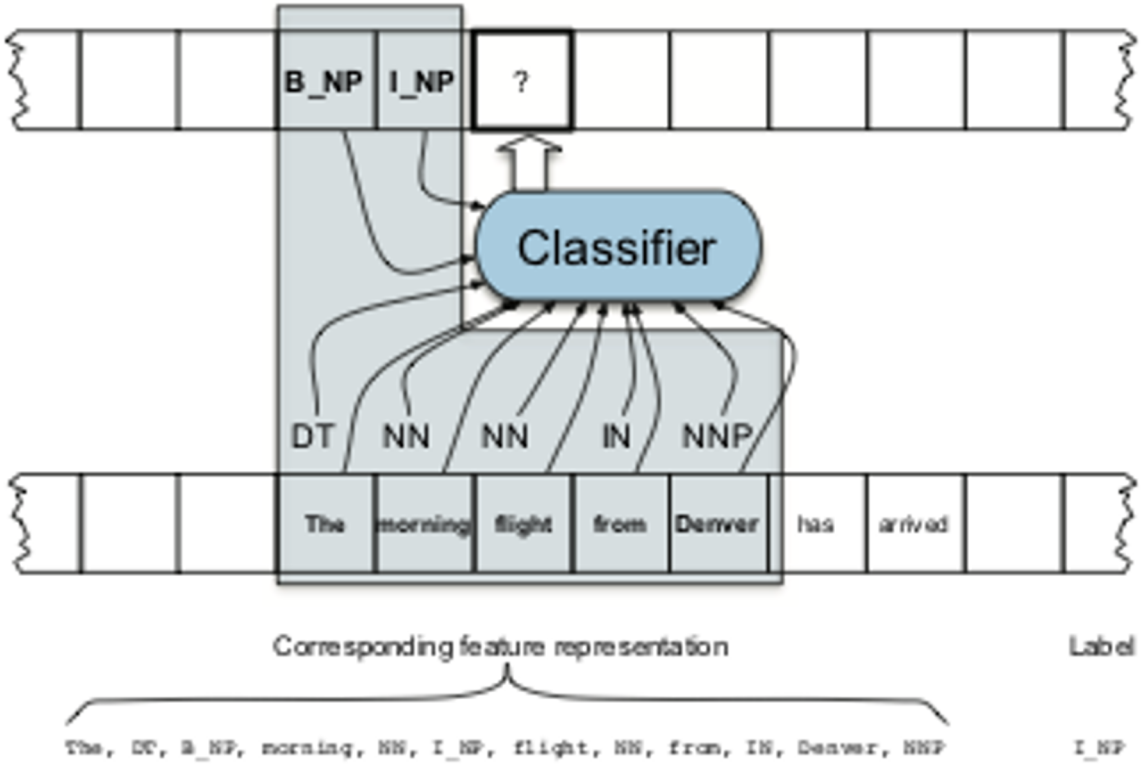
- Chunck parsing -> tagging
- *n* chuncks -> *2n+1* tags
- **B**egin , **I**nside, **O**utside

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

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



E se le CFG non ci piacciono?

Back in U.S.S.R.

-> grammatiche a dipendenze