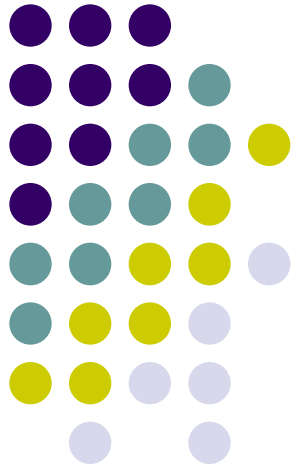
07.NLP-PARSING

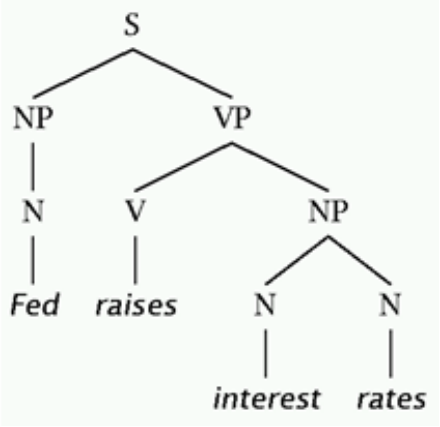
FIRDAUS SOLIHIN
UNIVERSITAS TRUNOJOYO



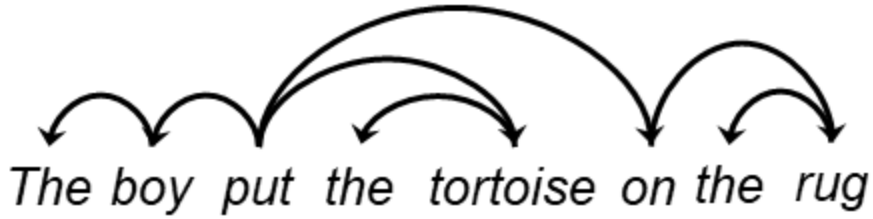
LINGUISTIC STRUCTURE

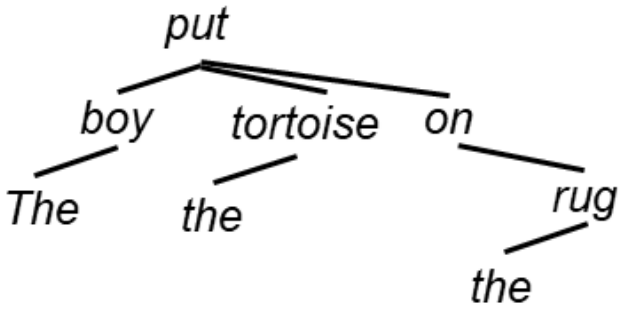
Struktur bahasa memiliki dua representasi

1. Constituency (phrase structure)

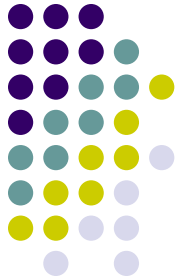


2. Dependency structure



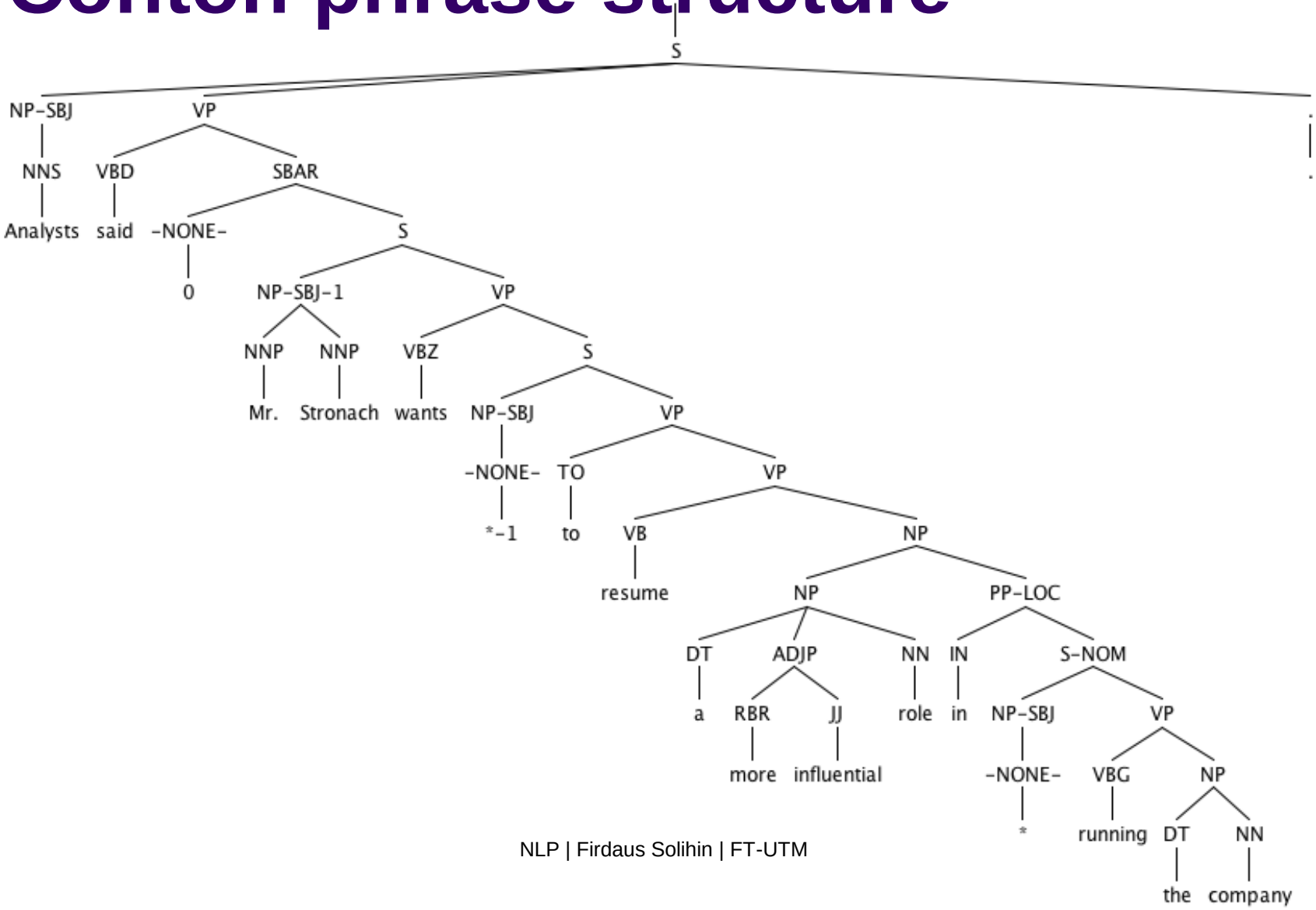


1. CONSTITUENCY (PHRASE STRUCTURE)

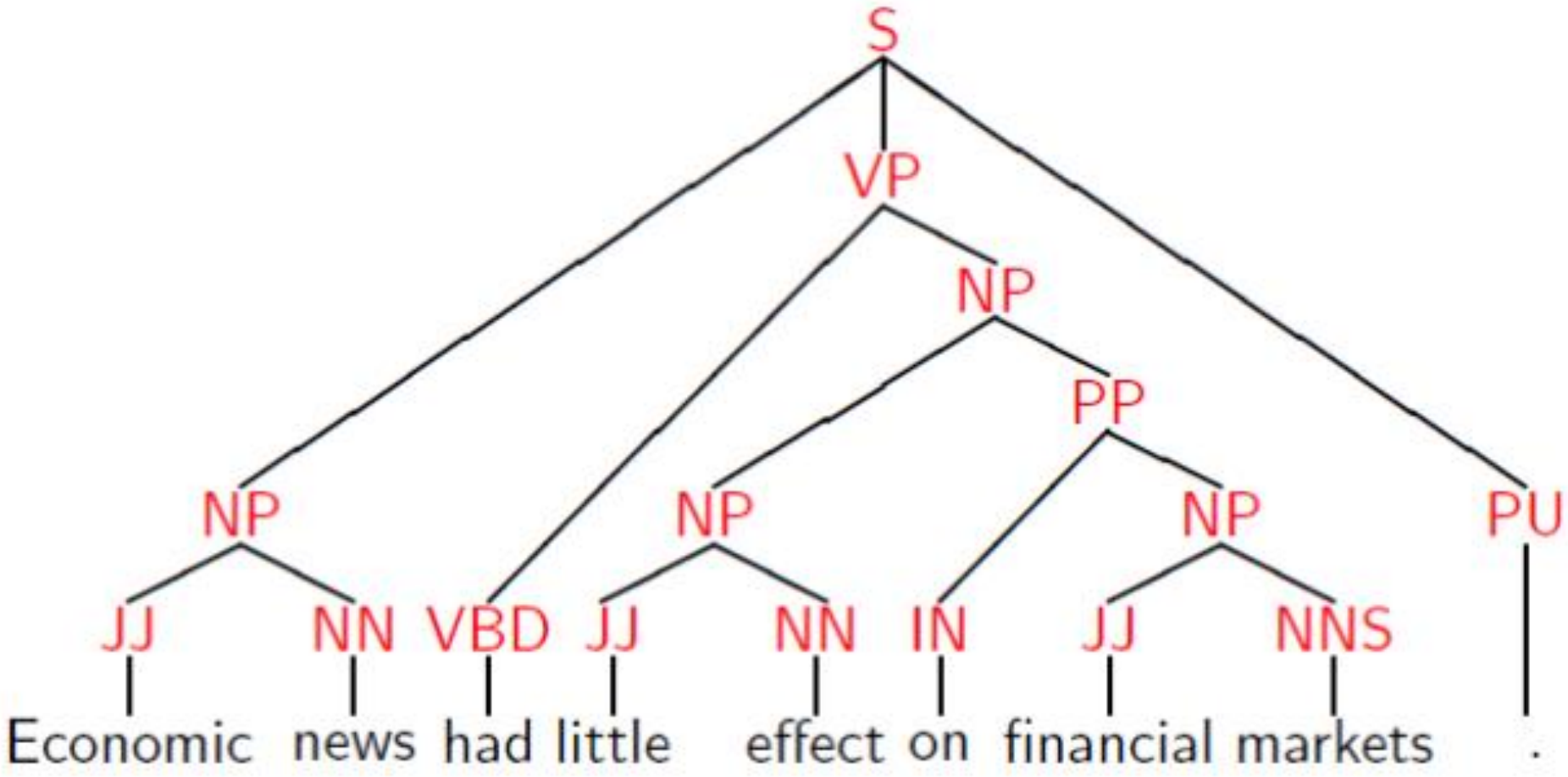


- phrase structure mengatur kata dalam urutan constituency yang bertingkat
- Bagaimana cara mengidentifikasi constituency
  - Distribution: dapat dipindahkan
    - Saya berbicara [kepada mahasiswa] [tentang narkoba]
    - Saya berbicara [tentang narkoba] [kepada mahasiswa]
  - Substitution/Expansion/Pro-form: dapat digantikan
    - Saya duduk [di atas kotak/tepat di atas kotak/di sana].
  - Coordination, regular internal structure, no intrusion, fragments, semantics,…

Contoh phrase structure

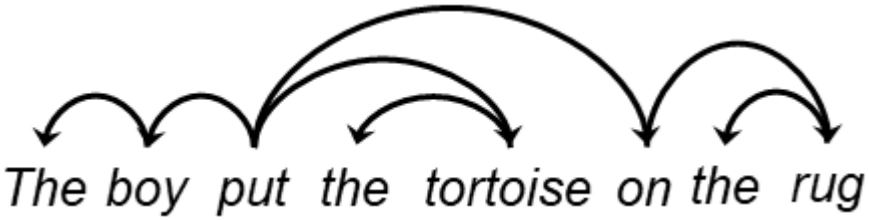


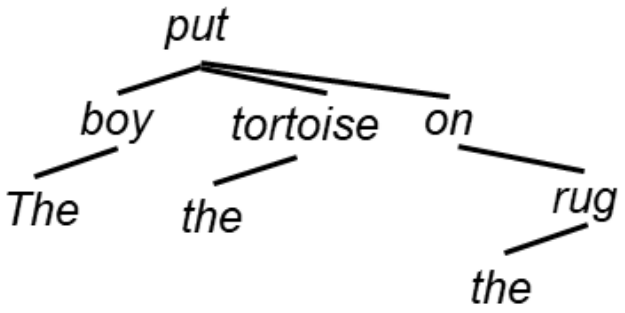
Contoh phrase structure



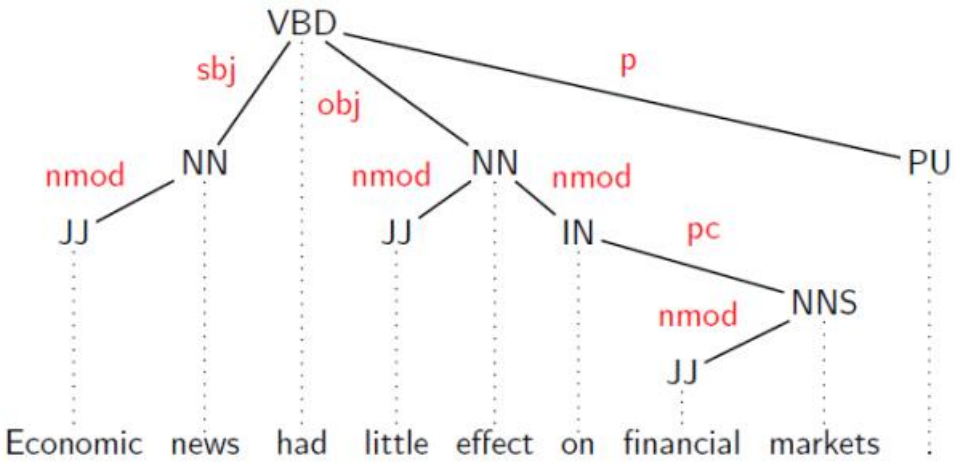
2. DEPENDENCY STRUCTURE

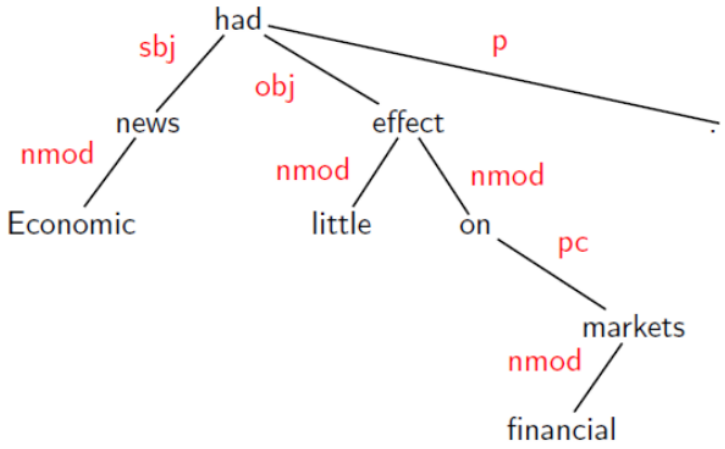
- Dependency structure atau struktur ketergantungan yaitu kata mana yang saling bergantung dengan kata lainnya

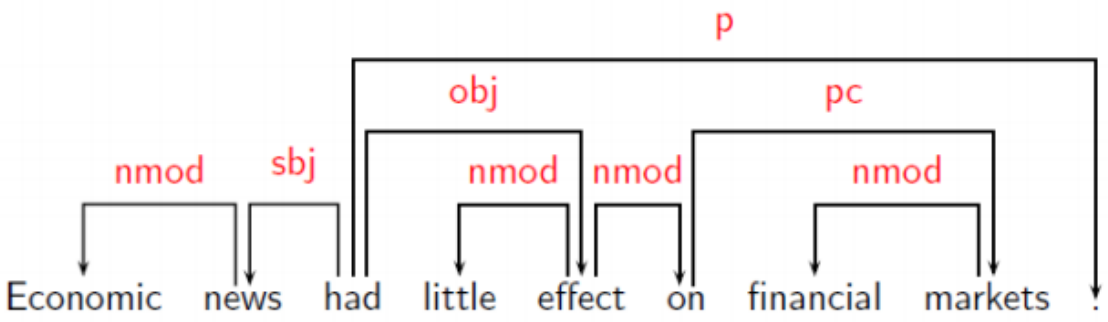


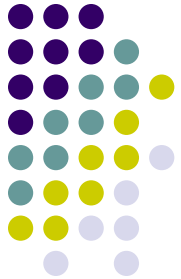


Contoh dependency structure



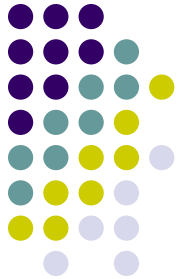






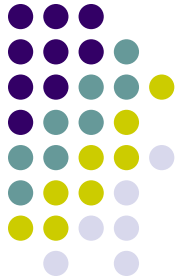
CONTEXT FREE GRAMMAR (CFG)

Context-Free Grammar



- Context-Free Grammar (CFG/Phrase-Structure Grammar/Backus-Naur Form) adalah suatu notasi matematis yang menyatakan aturan sebuah bahasa berdasarkan constituency.
- CFG terdiri dari dua bagian:
  1. Lexicon: daftar symbol/kata
  2. Sehimpunan (rewrite) rule atau production, yang menyatakan bagaimana symbol dalam bahasa dikelompokkan

Komponen dan Symbol CFG



- Komponen & symbol CFG terbagi 2 kelas:
  1. **Terminal symbols:** symbol yang merepresentasikan kata dalam bahasa (muncul dama string/kalimat). Daftar terminal symbol disebut juga Lexicon
  2. **Non-terminal symbols:** symbol yang merepresentasikan kelompok/aggregate/generalisasi terminal.
- Non-terminal yang diasosiasikan dengan terminal kadang disebut **preterminal symbol**. Pada NLP, preterminal = part of speech

KOMPONEN CFG

- **Terminal** = ditulis huruf kecil {a, λ, .. }
- **Non Terminal** = ditulis huruf besar {A, S=Start Here, …}
- Himpunan String / Kata yang terbentuk

Contoh CFG

- Himp = {λ,ab,aabb,aaabbb, …}
- CFG

  S → aSb

  S → λ

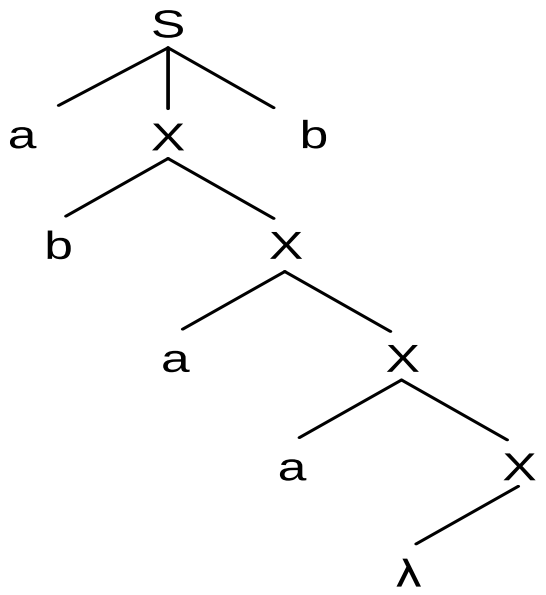
  Atau

  S → aSb | λ

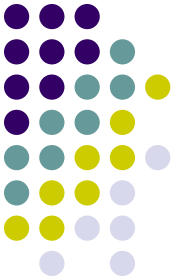
Struktur pohon (TREE)

- Struktur pohon (tree) dapat digunakan untuk menggambarkan proses pembentukan kata/string dari CFG
- $S \rightarrow a\ X\ b \mid b\ X\ a$

  $X \rightarrow a\ X \mid b\ X \mid \lambda$



Contoh CFG dlm NLP



Contoh *rule*:

Dalam bhs. Inggris, sebuah `NP` bisa terdiri dari `ProperNoun` atau `Determiner` diikuti `Nominal`. Sebuah `Nominal` bisa terdiri dari satu atau lebih `Noun`.

$NP \rightarrow Det\ Nominal$ $NP \rightarrow ProperNoun$ $Nominal \rightarrow Noun \mid Noun\ Nominal$

Contoh *lexicon*:

Lexicon bisa saja dinyatakan sebagai aturan context-free sebagai berikut:

$Det \rightarrow a$ $Det \rightarrow the$ $Noun \rightarrow flight$

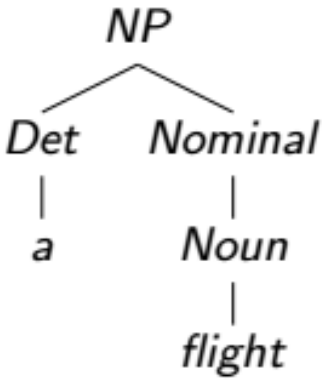
CFG sebagai pseudo-NLG

Sebagai generator:

CFG sebagai mesin yang menghasilkan kalimat sah sebagai output.

Sebagai generator, bacalah tanda "$\rightarrow$" sebagai: *"rewrite symbol on the left with the string of symbols on the right"*.

Mis, mulai dengan *NP*,
kita ganti *NP* dengan *Det Nominal*
lalu ganti *Nominal* dengan *Noun*,
lalu ganti *Det* dengan *a*,
dan ganti *Noun* dengan *flight*.



CFG bisa digunakan untuk menghasilkan semua kalimat yang bisa di-derive dari sebuah non-terminal, misalnya *NP*. Urutan pengaplikasian rule disebut sebagai derivation sebuah kalimat – biasanya direpresentasikan sebagai sebuah parse tree.

CFG sebagai pseudo-NLG

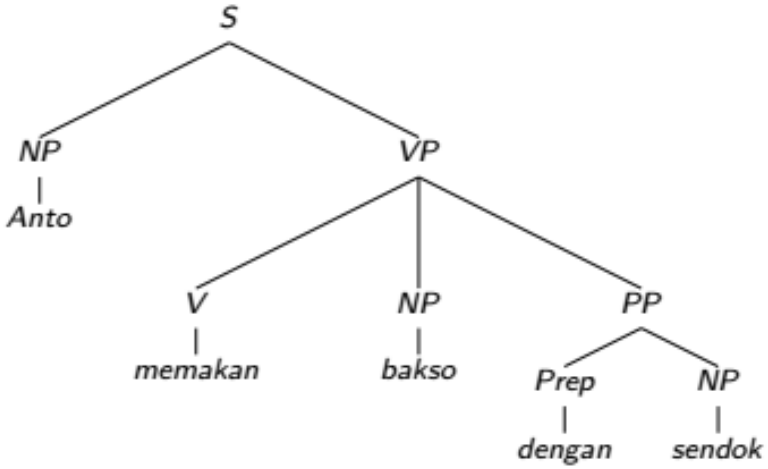
- Bayangkan kita membuat sebuah program yang inputnya sebuah CFG, dan secara random menghasilkan beberapa kalimat yang sah. Ini bisa saja dianggap contoh sistem NLG (Natural Language Generation) yang paling sederhana.
  - Random nonsense: http://www.elsewhere.org/pomo/ (The Postmodernism Generator)
  - Computer science paper generator :-) http://pdos.csail.mit.edu/scigen/ (SCIgen)
  - Ketika digabung dengan parsing, bisa membuat "conversational agent",mis. ELIZA (cf. Turing Test)
- Program-program ini mengetahui syntax, tetapi tidak semantics (makna).
- Sistem NLG "sesungguhnya" secara eksplisit menyatakan makna.

CFG sebagai parser

Sebagai parser:

CFG sebagai mesin yang menghasilkan struktur untuk sebuah kalimat input.

Sebagai parser, bacalah tanda "$\rightarrow$" sebagai: *"if you see the symbols on the right, rewrite with the symbol on the left"*.

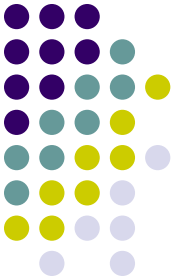


- Masalah bahasa manusia: satu kalimat, banyak struktur (=**ambiguity**/**kerancuan**)
- Tujuan akhir *parsing*: agar program dapat "memahami" **semantics**/**makna** dari kalimat.

Notasi alternatif, *bracketed notation*:

[$_S$ [$_{NP}$ Anto] [$_{VP}$ [$_V$ memakan] [$_{NP}$ bakso] [$_{PP}$ [$_{Prep}$ dengan] [$_{NP}$ sendok]]]]

Grammar singkat untuk bahasa Inggris (ATIS)



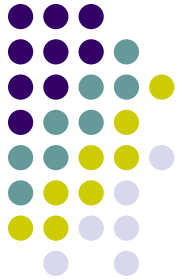
Lexicon untuk $\mathcal{L}_0$

Noun	→	*flights* | *breeze* | *trip* | *morning* | . . .
Verb	→	*is* | *prefer* | *like* | *need* | . . .
Adjective	→	*cheapest* | *first* | *other* | *direct* | . . .
Pronoun	→	*me* | *I* | *you* | *it* | . . .
ProperNoun	→	*Alaska* | *Baltimore* | *Chicago* | *Garuda* | . . .
Determiner	→	*the* | *a* | *an* | *this* | . . .
Preposition	→	*from* | *to* | *on* | *near* | . . .

Grammar untuk $\mathcal{L}_0$

S	→	*NP VP*	I + prefer a morning flight
NP	→	*Pronoun*	I
	|	*ProperNoun*	Los Angeles
	|	*Det Nominal*	a + flight
Nominal	→	*Noun Nominal*	morning + flight
	|	*Noun*	flights
VP	→	*Verb*	do
	|	*Verb NP*	prefer a morning flight
	|	*Verb NP PP*	leave Boston in the morning
	|	*Verb PP*	leave in the morning
PP	→	*Prep NP*	from Los Angeles

Grammaticality



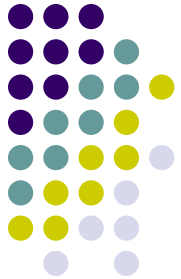
- Kalimat-kalimat yang bisa di-derive dari S dikatakan grammatical.
- Kalimat-kalimat yang **TIDAK** bisa di-derive dari S dikatakan **ungrammatical**.
- Perbedaan yang sangat “tajam” untuk bahasa formal seperti ini terkadang kurang cocok untuk bahasa natural/manusia . . .
- Dalam bidang linguistics, pemodelan ini disebut generative grammar (Chomsky), akhir '60-an.

Denisi formal CFG

Ingat, sebuah CFG adalah 4-tuple:

1. himpunan simbol non-terminal N
2. himpunan simbol terminal Σ (di mana $N \cap \Sigma = \emptyset$)
3. himpunan production rule P, masing-masing berbentuk $A \rightarrow \alpha$ di mana
   - $A \in N$, dan
   - $\alpha \in (\Sigma \cup N)*$ (dkl. α adalah string simbol terminal/nonterminal)
4. sebuah start symbol $S \in N$

Bahasa formal yang dinyatakan oleh sebuah CFG adalah himpunan string yang bisa di-*derive* dari symbol khusus: *start symbol (S)*. Dalam NLP, S sering diartikan sebagai *"sentence"*.



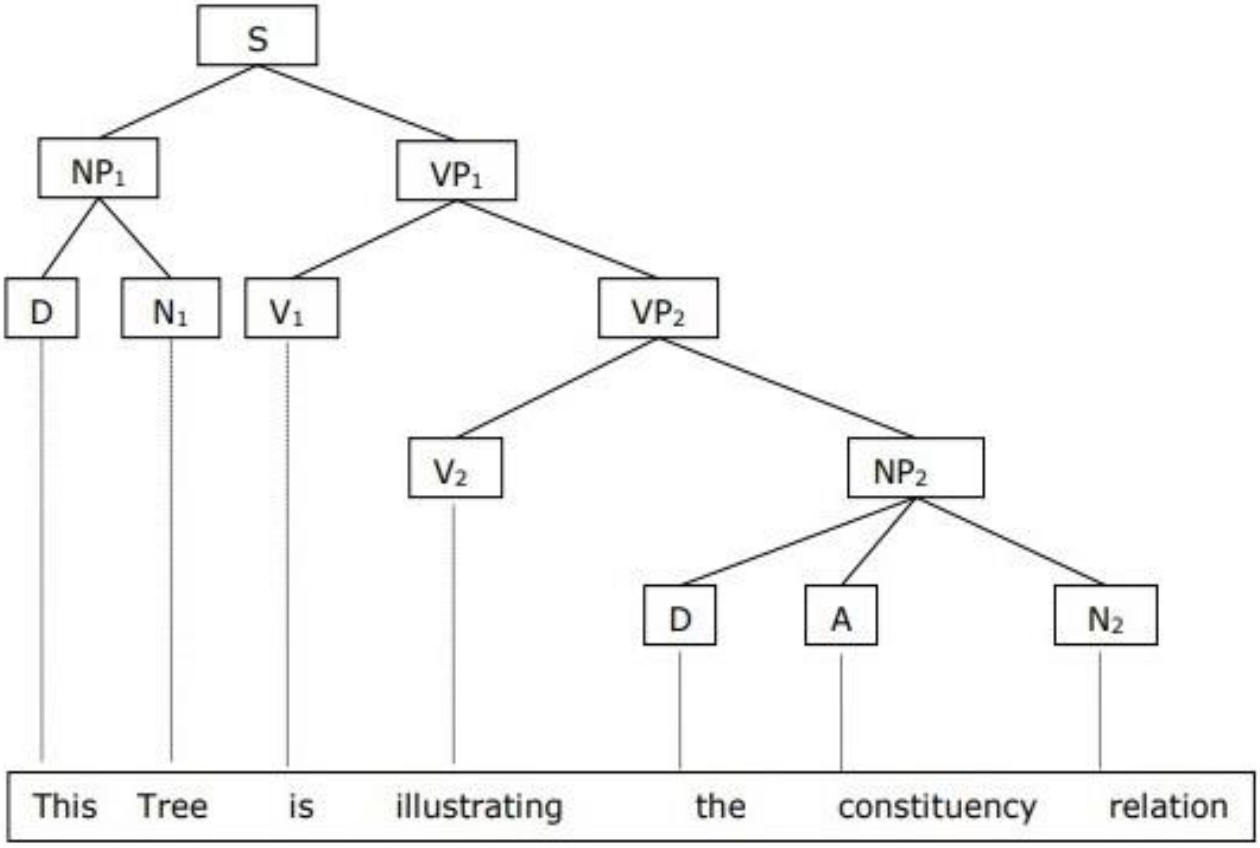
PARSING

Syntatic Parsing

- Memeriksa makna yang terkandung dalam text dan membandingkannya dengan aturan tata bahasa formal
- *Grammar* : aturan pembentuk suatu kalimat dalam sebuah bahasa dan dikenali dengan tata bahasa
- Proses searching:
  - *Context Free Grammar* (CFG)

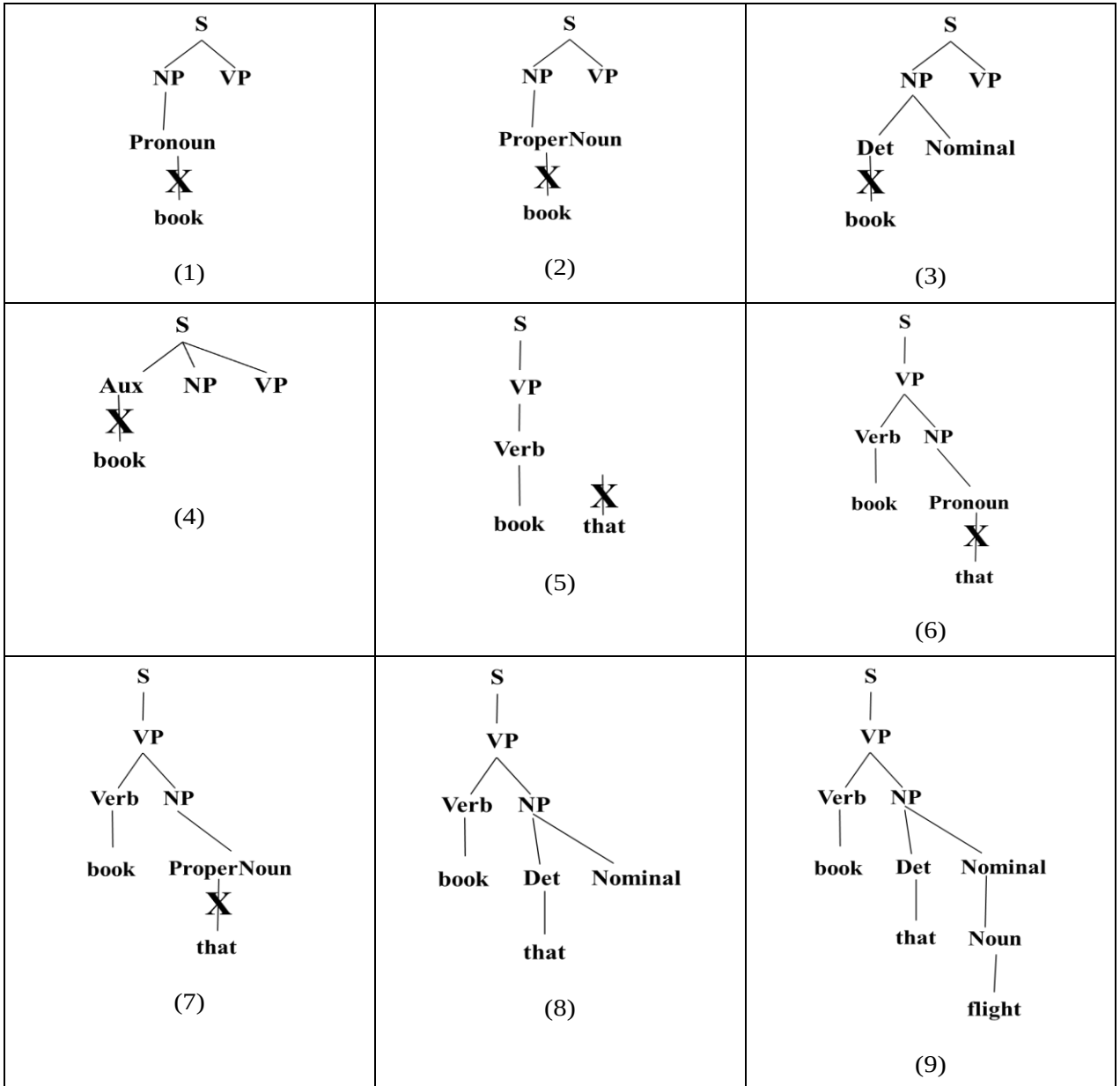
Grammar	Lexicon
S → NP VP	Det → the | a | that | this
S → Aux NP VP	Noun → book | flight | meal | money
S → VP	Verb → book | include | prefer
NP → Pronoun	Pronoun → I | he | she | me
NP → Proper-Noun	Proper-Noun → Houston | NWA
NP → Det Nominal	Aux → does
Nominal → Noun	Prep → from | to | on | near | through
Nominal → Nominal Noun	
Nominal → Nominal PP	
VP → Verb	
VP → Verb NP	
VP → VP PP	
PP → Prep NP	

Ilustrasi Parse Tree

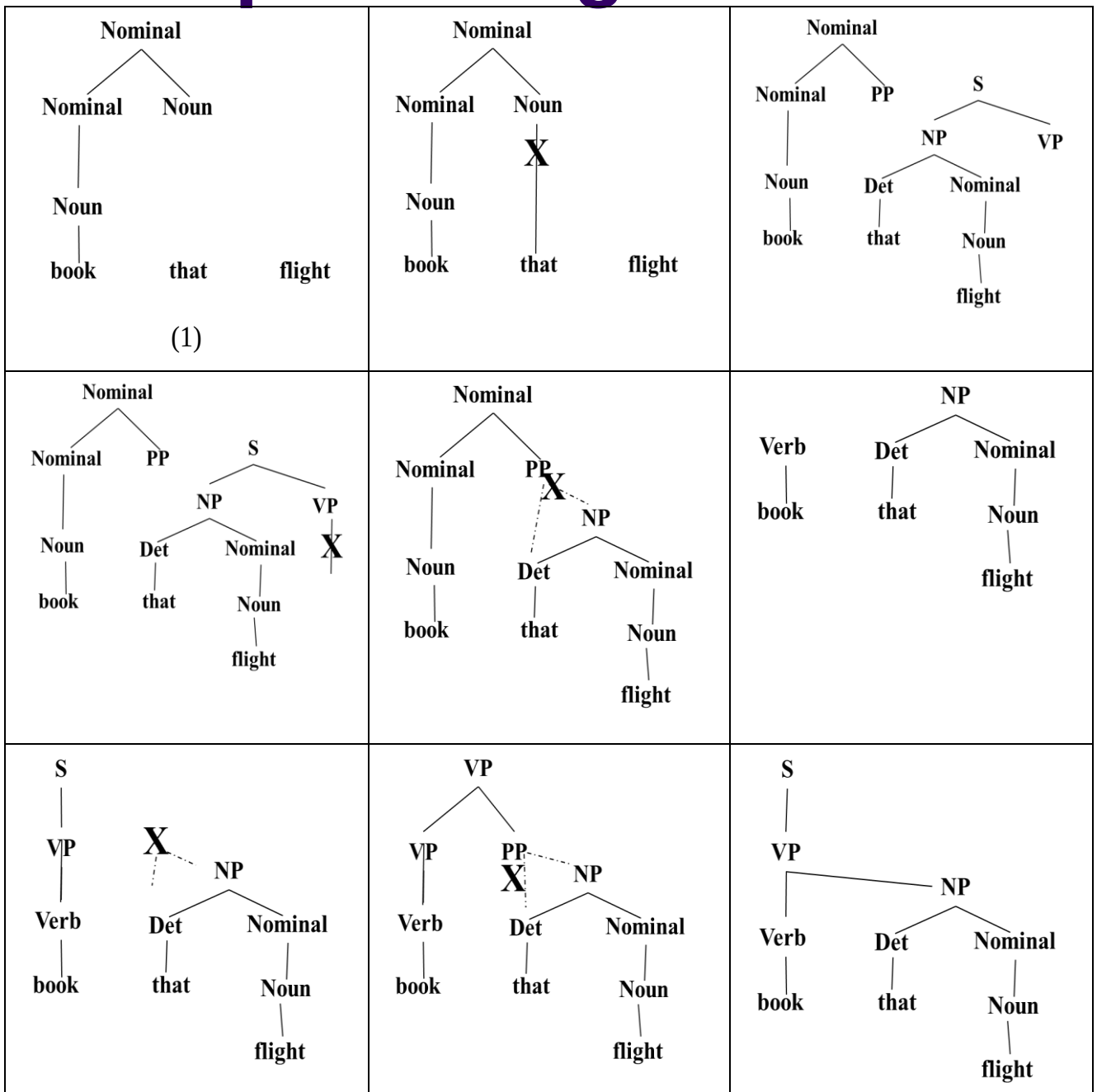


Top Down Parsing

- Kalimat : 'book that flight'

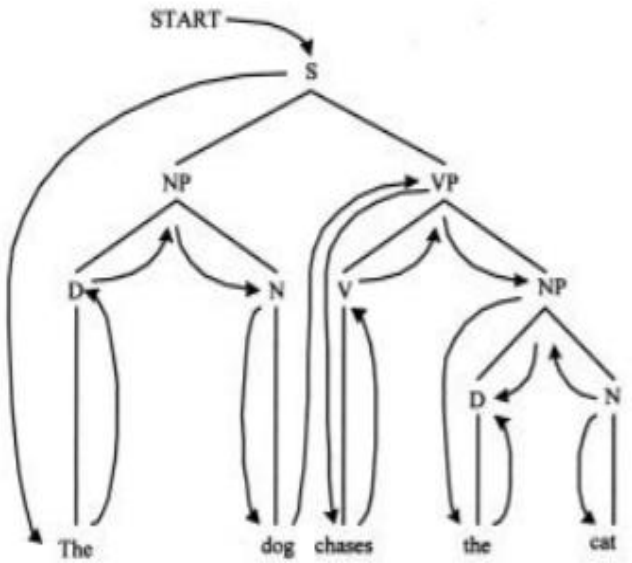


Bottom Up Parsing



Kelemahan

- Top Down parsing → tidak dapat menangani masalah *left-recursion*
- Bottom Up parsing → tidak dapat menangani grammar dengan *empty production*.
- Gabungan : model *Left-Corner* dan *Earley's Parsing*



Statistical Parsing

- Model sintaktis parsing tetapi dalam grammarnya terdapat nilai probabilitas
- Probabilistic Context Free Grammar (PCFG)

Grammar	Prob	
S → NP VP	0.8	
S → Aux NP VP	0.1	+ 1.0
S → VP	0.1	
NP → Pronoun	0.2	
NP → Proper-Noun	0.2	+ 1.0
NP → Det Nominal	0.6	
Nominal → Noun	0.3	
Nominal → Nominal Noun	0.2	+ 1.0
Nominal → Nominal PP	0.5	
VP → Verb	0.2	
VP → Verb NP	0.5	+ 1.0
VP → VP PP	0.3	
PP → Prep NP	1.0	

Lexicon

Det → the | a | that | this
0.6 0.2 0.1 0.1

Noun → book | flight | meal | money
0.1 0.5 0.2 0.2

Verb → book | include | prefer
0.5 0.2 0.3

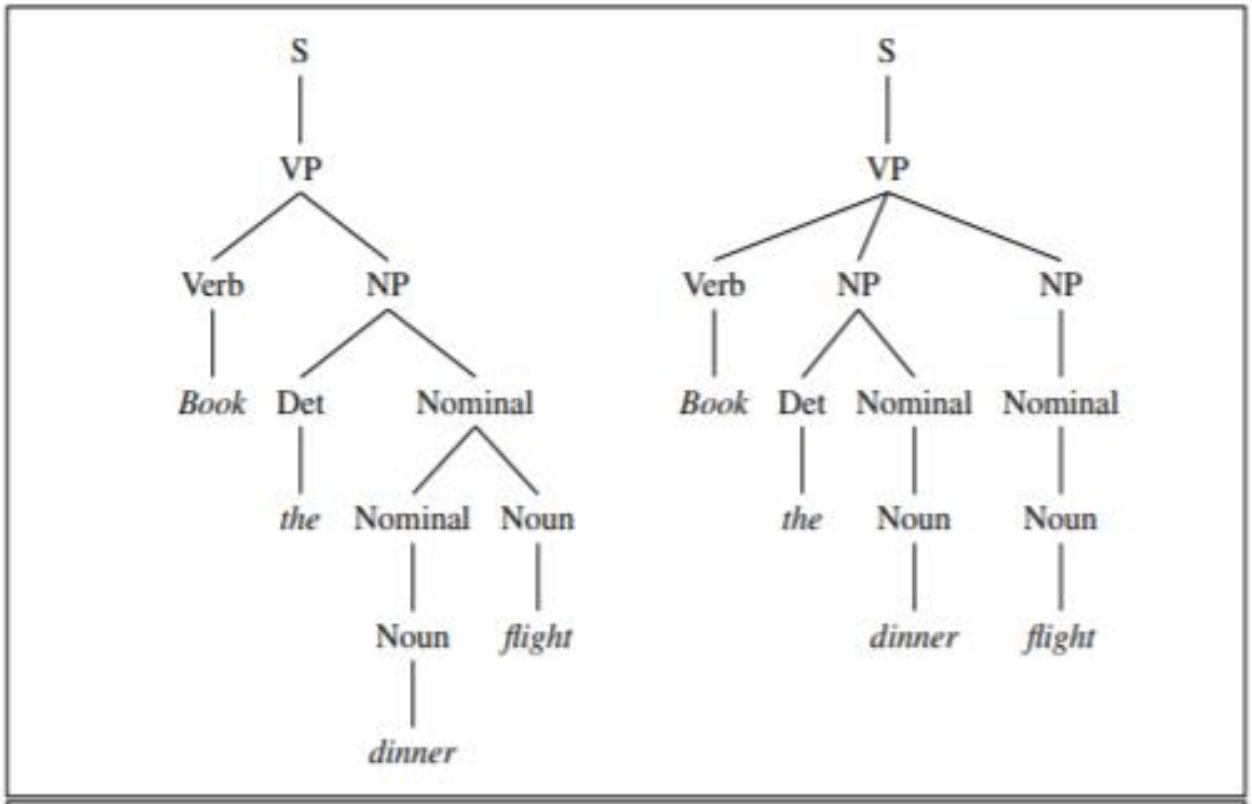
Pronoun → I | he | she | me
0.5 0.1 0.1 0.3

Proper-Noun → Houston | NWA
0.8 0.2

Aux → does
1.0

Prep → from | to | on | near | through
0.25 0.25 0.1 0.2 0.2

Contoh



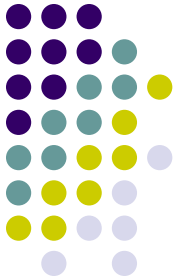
	Rules		P		Rules		P
S	→	VP	.05	S	→	VP	.05
VP	→	Verb NP	.20	VP	→	Verb NP NP	.10
NP	→	Det Nominal	.20	NP	→	Det Nominal	.20
Nominal	→	Nominal Noun	.20	NP	→	Nominal	.15
Nominal	→	Noun	.75	Nominal	→	Noun	.75
				Nominal	→	Noun	.75
Verb	→	book	.30	Verb	→	book	.30
Det	→	the	.60	Det	→	the	.60
Noun	→	dinner	.10	Noun	→	dinner	.10
Noun	→	flight	.40	Noun	→	flight	.40

Dependency Parsing

- Model parsing dengan memperhatikan konsep keterkaitan atau hubungan antar kata dalam kalimat
- Contohnya:

Kalimat 1 : I’ve never been afraid
Dependency Parser : nsubj (afraid-5, i-1)
aux (afraid-5, ‘ve-2)
neg (afraid-5, never-3)
cop (afraid-5, been-4)
root (ROOT-0, afraid-5)
POS Tagging : I’ve|NNP never|RB been|VBN afraid|JJ

Referensi



- Buku Rerensi Utama Kuliah
- Natural Language Processing, Lecture Slides by Dan Jurafsky and Christopher Manning, Stanford Coursera course
- Komputasi Bahasa Alami, Dr. Fika Hastarita Rachman, S.T., M.Eng. Jurusan Teknik Informatik, Universitas Trunojoyo Madura
  2020
- CSC4602354 Pengolahan Bahasa Manusia, Rahmad Mahendra, Fakultas Ilmu Komputer, Universitas Indonesia