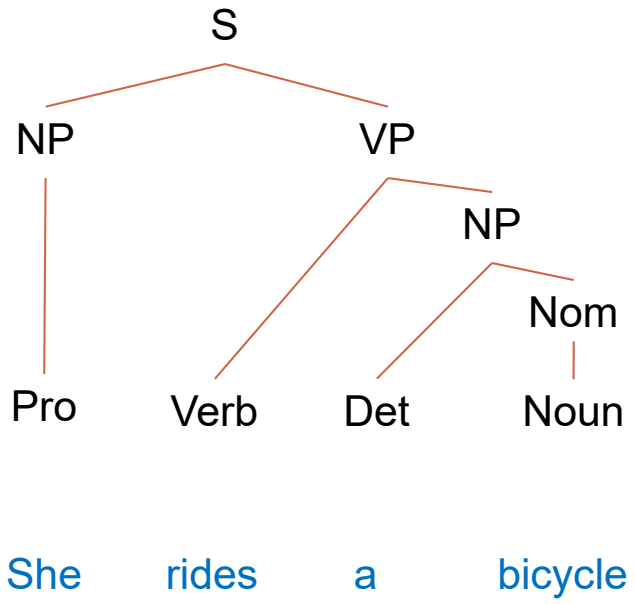
Natural Language Processing

Lecture #5
Syntactical Analysis

Hutchatai Chanlekha

Department of Computer Engineering, Kasetsart University

- From the following sentences:
  - She rides a bicycle.
  - My friend gave me a book.
- What are subject, verb, and object of the sentences?
- What are the syntactic structure of these sentences?



Parser

Syntax

- Constituency
- Grammatical relations
- Subcategorization and dependencies
  - Can be modeled by grammars that are based on context-free grammars.
    - Context-free grammars are thus the backbone of many models of the syntax of NL.

Constituency

- Constituent
  - Groups of words may behave as a single unit or phrase
    - สุดา, ผู้หญิงผมยาว, พระที่นั่งอนันตสมาคม, หนังสือภาษาอังกฤษ, ฯลฯ
    - Michael, the house, Russian Hill, three-story structure, etc.
- Why do we want to know what are constituents in the sentence?
  - *"The people who live on that lovely village are very friendly"*
- How do we know that these words group together, i.e. form a constituent?

Constituency (cont.)

- Evidence for constituency
  - Syntactic information
    - Three parties from Brooklyn attracts …
    - A high-class sop such as Mindy's arrives …
    - The Broadway coppers love …
    - They sit …
  - One possible conclusion from the example:
    - "Noun phrases can occur before verbs"

Constituency (cont.)

- Evidence for constituency
  - Preposed or postposed constructions
    - A phrase, such as prepositional phrase …
      - The entire phrase can be placed in a number of different locations in a sentence, but the individual words making up the phrase cannot:
      - "On July seventeenth"
        - On July seventeenth, I'd like to fly from Bangkok to Tokyo
        - I 'd like to fly on July seventeenth from Bangkok to Tokyo
        - I 'd like to fly from Bangkok to Tokyo on July seventeenth
        - On July, I'd like to fly seventeenth from Bangkok to Tokyo
        - On I'd like to fly July seventeenth from Bangkok to Tokyo
        - I'd like to fly on July from Bangkok to Tokyo seventeenth

Context-free rules and trees

- The most commonly used mathematical system for modeling constituent structure in NL is Context-free Grammar (CFG)
  - Also called Phrase-Structure Grammar
  - The formalism is equivalent to Backus-Naur Form (BNF)
- CFG
  - A device for generating sentences
  - A device for assigning a structure to a given sentence

Context-free rules and trees

- CFG consists of set of rules and productions, and lexicon
  - Lexicon
    - Words or symbols
    - For example:

Det	→	a | the
Noun	→	flight

  - Rules and productions
    - Express the ways that language symbols can be grouped or ordered together
    - For example: productions of NP

Nominal	→	Noun | Noun Nominal
NP	→	Det Nominal
NP	→	ProperNoun

  - CF rules can be hierarchically embedded
    - Can combine the previous rule with others, such as combine the above rules with rules that express facts about the lexicon

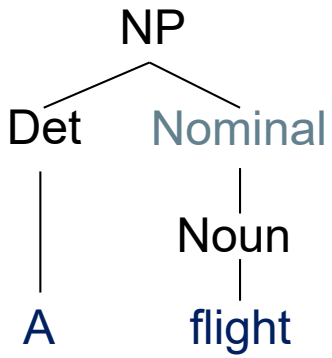
Context-free rules

- Symbols used in CFG
  - Terminal symbols
    - Symbols corresponding to words in the language
    - Lexicon is the set of rules that introduce terminal symbols
  - Non-terminal symbols
    - Symbols expressing clusters or generalizations of terminal symbols
  - In CF rule

    LHS → RHS
    - Item to the left of ' → '
      - Single non-terminal symbol expressing some cluster or generalization
    - Item to the right of ' → '
      - Ordered list of one or more terminals and non-terminals

Context-free Grammar

- → : rewrite the symbol on the left with the string of symbols on the right
  - From the rules, start with NP
    - NP → Det Nominal
    - Nominal → Noun
    - Det → a | the
    - Noun → flight



'a flight' can be derived from the non-terminal NP

Context-free Grammar

- Formal language defined by CFG is the set of strings that are derivable from the ***start symbol***.
- Each grammar must have one designated start symbol.
  - Often called ***S***
  - Since CFG are often used to define sentences
    - ***S*** is usually interpreted as "sentence"
- Example of CFG

S	→	NP VP	(such as 'I prefer a morning flight')
VP	→	Verb NP	(such as 'prefer a morning flight')
VP	→	Verb NP PP	(such as 'leave Boston in the morning')
VP	→	Verb PP	(such as 'leaving on Thursday')
PP	→	Preposition NP	(such as 'from Bangkok')
NP	→	Det Nominal	(such as 'a morning flight')
Nominal	→	Noun | Noun Nominal	(such as 'morning flight')

Example: Lexical for $\mathcal{L}_0$

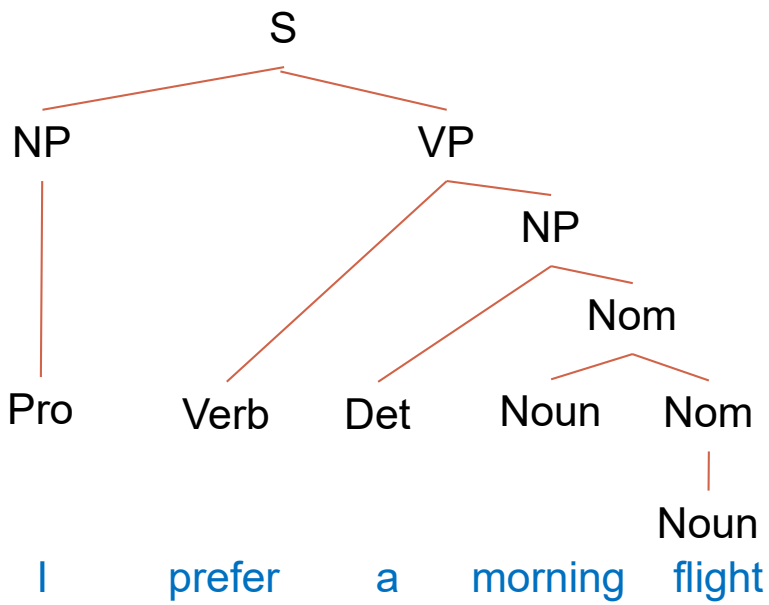
Noun	→	flights | breeze | trip | morning | …
Verb	→	is | prefer | like | need | want | fly | …
Adjective	→	cheapest | non-stop | first | latest | other | direct | …
Pronoun	→	me | I | you | it
Proper-Noun	→	Alaska | Baltimore | Los Angeles | Chicago | United | American
Determiner	→	the | a | an | this | these | that | …
Preposition	→	from | to | on | near | …
Conjunction	→	and | or | but

Example : grammar for $\mathcal{L}_0$

S	→	NP VP	I + want a morning flight
NP	→	Pronoun	I
		| Proper-noun	Los Angeles
		| Det Nominal	a + flight
Nominal	→	Noun Nominal	morning + flight
		| Noun	flight
VP	→	Verb	do
		| Verb NP	want + a flight
		| Verb NP PP	leave + Boston + in the morning
		| Verb PP	leaving + on Thursday
PP	→	Preposition NP	from + Los Angeles

Notation

- Parse tree



- Bracket notation

$[_S\ [_{NP}\ [_{Pro}\ \textbf{I}\ ]\ ]\ [_{VP}\ [_V\ \textbf{prefer}]\ [_{NP}\ [_{Det}\ \textbf{a}\ ]\ [_{Nom}\ [_N\ \textbf{morning}]\ [_{NOM}\ [_N\ \textbf{flight}]]]]]]$

- Let’s try writing parse tree
  - http://nlp.stanford.edu:8080/parser/
  - http://lfg-demo.computing.dcu.ie/lfgparser.html

CFG defines a Formal language

- Grammatical sentences
  - Sentences, which can be derived by a grammar, are in the formal language defined by that grammar
- Ungrammatical sentences
  - Sentences, which can NOT be derived by a given formal grammar, are NOT in the language defined by that grammar
- In linguistics, the use of formal languages to model NL is called “Generative Grammar”
  - Language is defined by a set of possible sentences generated by the grammar

Summary of CFG

- CFG is a 4-tuple (N, $\sum$, α, S)
  - A set of non-terminal symbols *N*
  - A set of terminal symbols $\sum$
  - A set of productions *P*, each of the form *A* → α
    - A is a non-terminal symbol
    - α is a string of symbols from the infinite set of strings $(\sum \cup N)^*$
  - A designated start symbol *S*

Application of grammar rewrite rules

S → NP VP
S → Aux NP VP
S → VP
NP → Det NOM
NOM → Noun
NOM → Noun NOM
VP → Verb
VP → Verb NP

Det → *that* | *this* | *a* | *the*
Noun → *book* | *flight* | *meal* | *man*
Verb → *book* | *include* | *read*
Aux → *does*

S → NP VP
→ Det NOM VP
→ The NOM VP
→ The Noun VP
→ The man VP
→ The man Verb NP
→ The man read NP
→ The man read Det NOM
→ The man read this NOM
→ The man read this Noun
→ The man read this book

Phrase structure of English

- Sentence
  - Declarative
  - Imperative
    - Simple example
      - S → VP (“ทำการบ้าน”, “Give me a book.”)
  - Question
    - Yes-no-question, WH-question
    - Simple example
      - S → Aux NP VP (“Did you sleep last night?”)
      - etc.
      - wh-phrase (who, where, what, which, how, why)
        - S → wh-NP VP (What airlines fly from Bangkok to Tokyo?)
        - S → wh-NP Aux NP VP (What food do you want to eat?)
        - etc.
  - Etc.

Phrase structure of English (cont.)

- **Phrase**
  - A group of words or a single word that forms a constituent
    - Functions as a single unit in the syntax of a sentence
  - Consist of head and dependent
    - Head word
      - Key word that identifies the type and linguistic features of the phrase
      - Characterize the primary grammatical role of the phrase
    - Dependents
      - Non-head word, usually modify the head word
  - Common phrase types
    - too **slowly** — Adverb phrase (AdvP); the head is an adverb
    - very **happy** — Adjective phrase (AP); the head is an adjective
    - the big **balloon** — Noun phrase (NP); the head is a noun
    - **on** the table — Preposition phrase (PP); the head is a preposition
    - **watch** television — Verb phrase (VP); the head is a verb

Grammar and Parsing

- Context-free grammars are a declarative formalism
  - Do not specify how the parse tree for a given sentence should be computed
- Run the grammar backwards to find the structure.
- Parsing can be viewed as a search problem.
  - Search through the space of all possible parse trees to find the correct parse tree
  - Search space of possible parse trees is defined by the grammar
- Goal of parsing search
  - Find all trees whose root is the start symbol, which match exactly the words in the input
- Two kinds of constraints that should help guide the search
  - Constraints from the data, i.e. input sentence itself
  - Constraints from the grammar
- Parsing strategies
  - Top-down (goal-directed search)
  - Bottom-up (data-directed search)

Recognizers and Parsers

- A recognizer
  - A program for which a given grammar and a given sentence returns YES if the sentence is accepted by the grammar(i.e., the sentence is in the language), and NO otherwise.
- A parser
  - In addition to doing the work of a recognizer also returns the set of parse trees for the string.

Soundness and Completeness

- A parser is sound if every parse it returns is valid/correct.
- A parser is complete if for any given grammar and sentence it is sound, produces every valid parse for that sentence, and terminates.
- For many cases, we settle for sound but incomplete parsers, e.g. return k-best parse trees

Example of miniature Grammar and Lexicon

Grammar	Lexicon
S → NP VP S → Aux NP VP S → VP NP → Det Nominal NP → Proper-Noun Nominal → Noun Nominal → Noun Nominal Nominal → Nominal PP VP → Verb VP → Verb NP	Det → the | this | a Noun → book | flight | meal | money Verb → book | include | prefer Aux → does Prep → from | to | on Proper-noun → Houston | TWA

Top-Down parsing

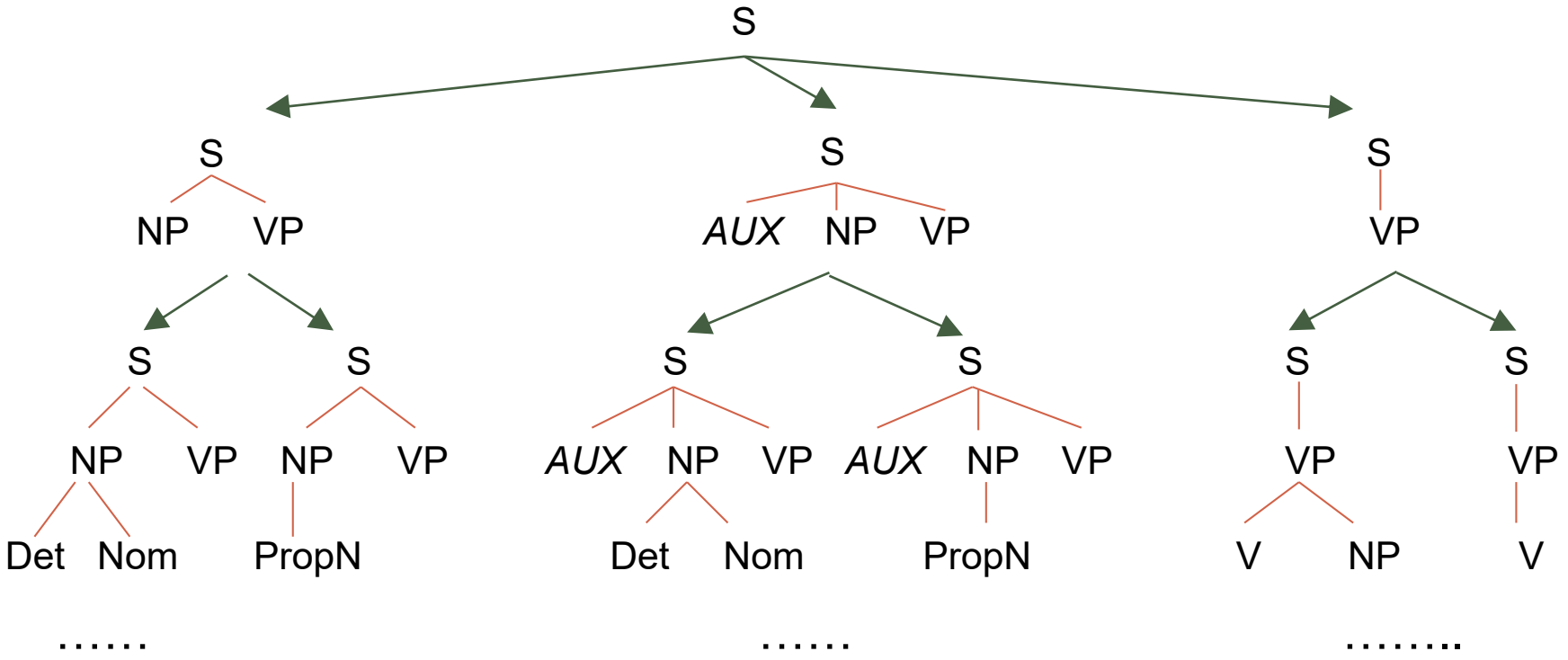
- Build from the root node to the leaves

- A top-down parser starts with a list of constituents to be built.
  - It rewrites the goals in the goal list by matching one against the LHS of the grammar rules, and expanding it with the RHS
  - Attempting to match the sentence to be derived.
- Trees are grown downward until they eventually reach the POS categories at the bottom of the tree.
  - Trees whose leaves fail to match all the words in the input can be rejected.

Top-down parsing: Example

- Parse: “Book that flight”

Grammar		Lexicon
S → NP VP	S → Aux NP VP	Det → the | this | a
S → VP		Noun → book | flight | meal | money
NP → Det Nominal	NP → Proper-Noun	Verb → book | include | prefer
Nominal → Noun	Nominal → Nominal PP	Aux → does
Nominal → Noun Nominal		Prep → from | to | on
VP → Verb	VP → Verb NP	Proper-noun → Houston | TWA



Problem with Top-down parsing

- Advantage of top-down parsing
  - Never explores subtrees that cannot find a place in some S-rooted tree
- Disadvantages
  - Left recursive rules
    - e.g. NP → NP PP lead to infinite recursion
  - Will do badly if there are many different rules for the same LHS.
    - Consider if there are 600 rules for S, 599 of which start with NP, but one of which starts with a V, and the sentence starts with a V.
  - Useless work: expands things that are possible top-down but are not consistent with the input.
  - Top-down is hopeless for rewriting parts of speech (pre-terminals) with words (terminals).
    - In practice, that is always done bottom-up as lexical lookup.
  - Repeated work: any where there is common sub-structure.

Bottom-up parsing

- The parse is successful if the parser succeeds in building a tree rooted in the start symbol that covers all of the input.

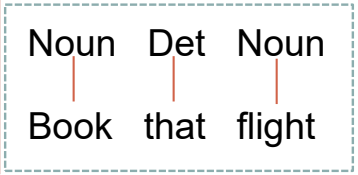
- Starting from a string to be parsed
  - If a sequence in the goal list matches the RHS of a rule, then this sequence may be replaced by the LHS of the rule.
  - Parsing is finished when the goal list contains just the start symbol.

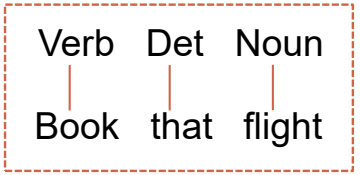
Bottom-up parsing: Example

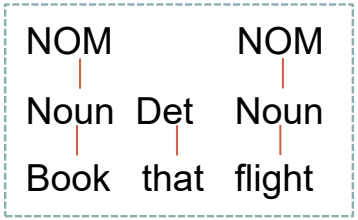
Lexicon

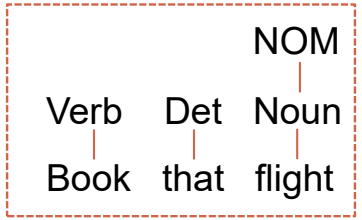
Det → the | this | a
Noun → book | flight | meal | money
Verb → book | include | prefer
Aux → does
Prep → from | to | on
Proper-noun → Houston | TWA

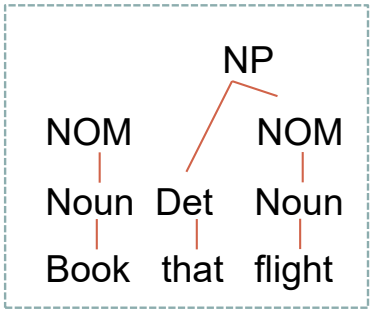
Book that flight

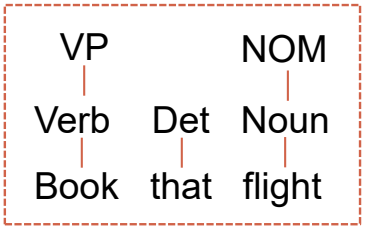


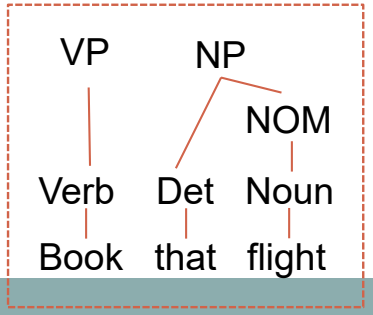


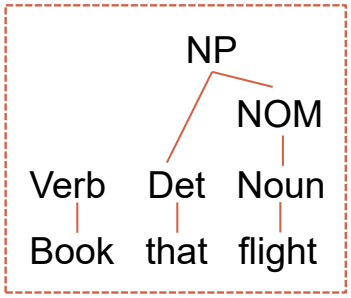


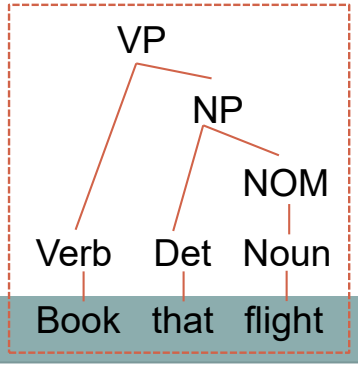












Grammar

S → NP VP
S → Aux NP VP
S → VP
NP → Det Nominal
NP → Proper-Noun
Nominal → Noun
Nominal → Nominal PP
Nominal → Noun Nominal
VP → Verb
VP → Verb NP

Problem with Bottom-up parsing

- Advantage
  - Never suggest trees that are not locally grounded in the actual input
- Disadvantage
  - Trees that have no hope of leading to an S, or fitting in with any of their neighbors, are generated with wild abandon.

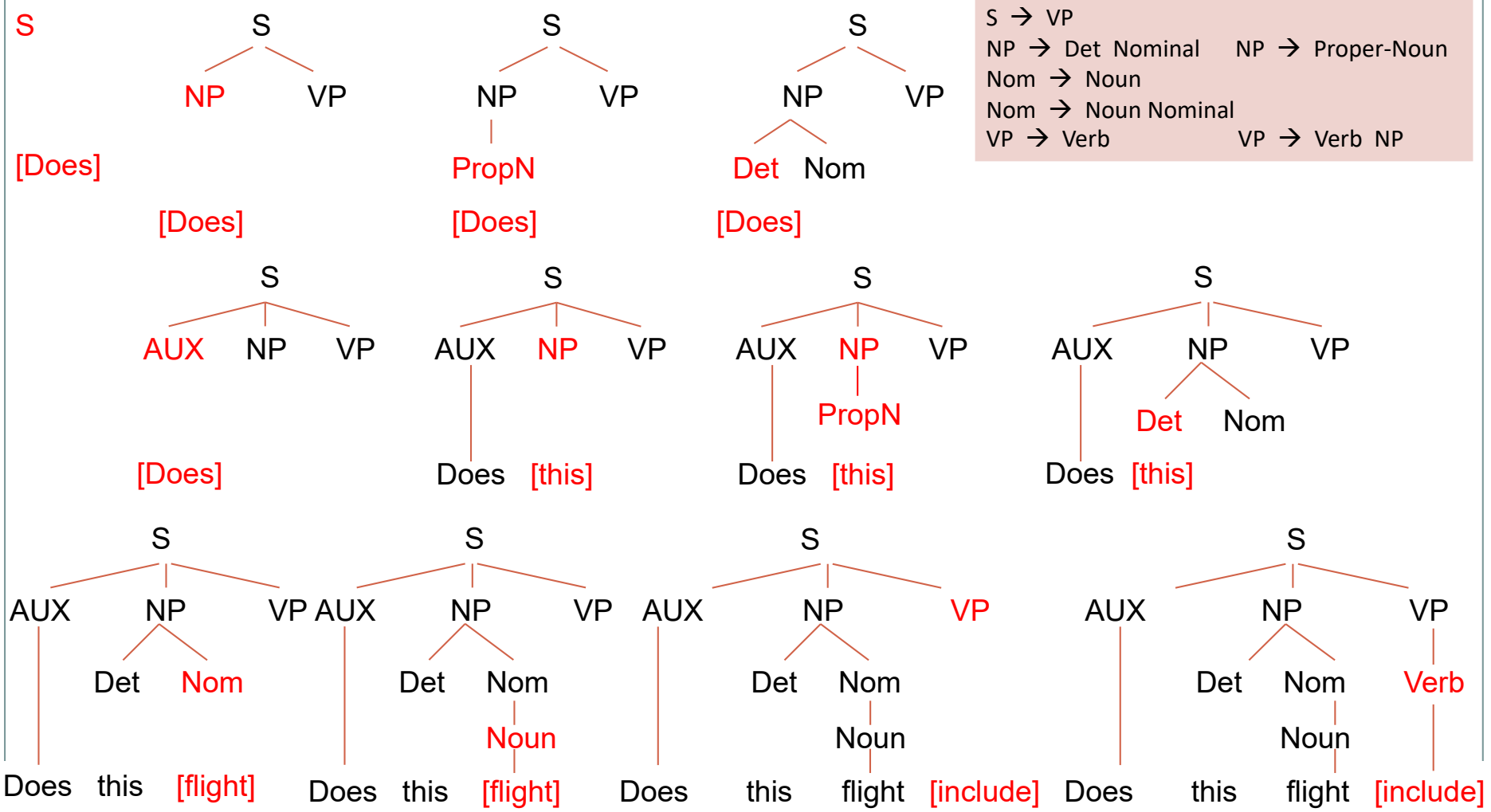
Combining Top-down and Bottom-up

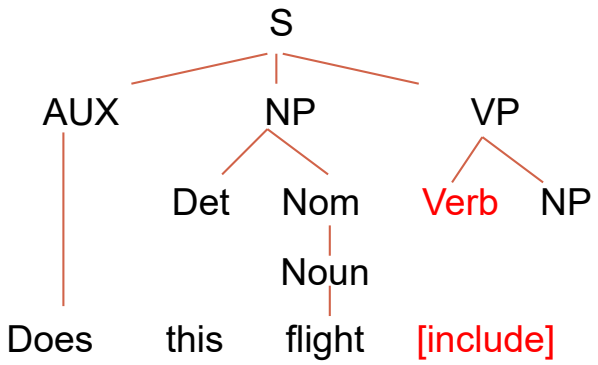
- Combining the best features of top-down and bottom-up parsing
  - Fairly straightforward
    - Adopt one technique as the primary control strategy used to generate trees (here, selecting top-down as primary control)
    - And use constraints from the other technique to filter out inappropriate parses on the fly (here, selecting bottom-up as filtering)

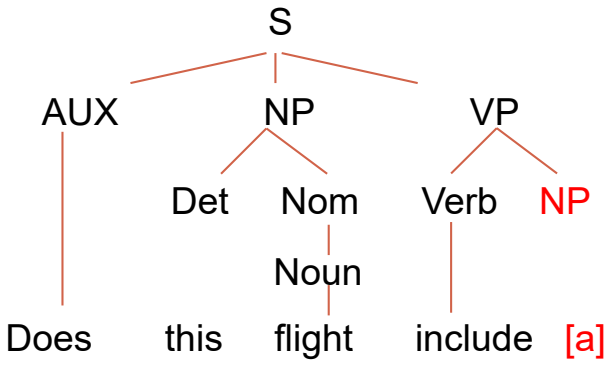
Example of top-down, depth-first, left-to-right derivation

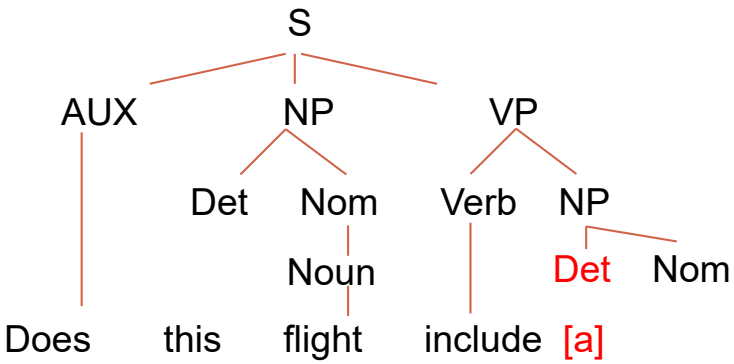
"Does this flight include a meal?"

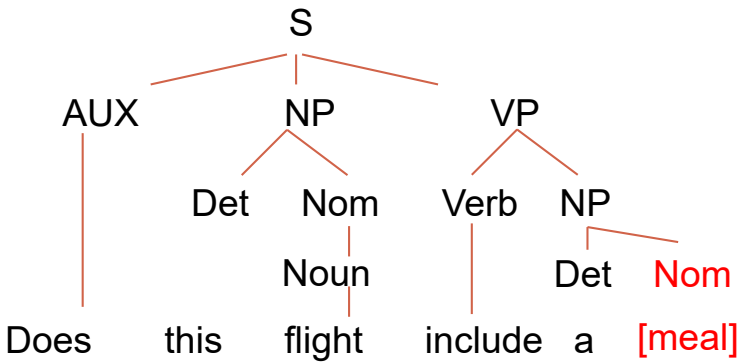
Grammar	
S → NP VP	S → Aux NP VP
S → VP	
NP → Det Nominal	NP → Proper-Noun
Nom → Noun	
Nom → Noun Nominal	
VP → Verb	VP → Verb NP

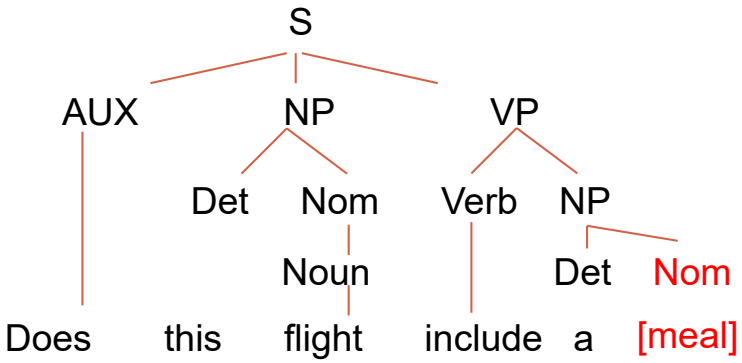



S
AUX
NP
VP
Det
Nom
Verb
NP
Noun
Does
this
flight
[include]


S
AUX
NP
VP
Det
Nom
Verb
NP
Noun
Does
this
flight
include
[a]


S
AUX
NP
VP
Det
Nom
Verb
NP
Noun
Det
Nom
Does
this
flight
include
[a]


S
AUX
NP
VP
Det
Nom
Verb
NP
Noun
Det
Nom
Does
this
flight
include
a
[meal]


S
AUX
NP
VP
Det
Nom
Verb
NP
Noun
Det
Nom
Does
this
flight
include
a
[meal]

Adding Bottom-Up Filtering

- Abandon any grammar rule if the current input cannot serve as the first word along the left edge of some derivation
  - the first word along the left edge of a derivation → Left corner
  - B is a left corner of A if $A \overset{*}{\Rightarrow} B\alpha$
  - B can be a left-corner of A if there is a derivation of A that begins with B
- Using left-corner table

Grammar	
S → NP VP	S → Aux NP VP
S → VP	
NP → Det Nominal	NP → Proper-Noun
Nominal → Noun	Nominal → Nominal PP
Nominal → Noun Nominal	
VP → Verb	VP → Verb NP

Category	Left corners
S	Det, Proper-noun, Aux, Verb
NP	Det, Proper-noun
Nominal	Noun
VP	Verb

- Example: "Does this flight include a meal?"
  - S → NP VP S → Aux NP VP S → VP
  - With left-corner notion, only *S → Aux NP VP* is a viable candidate

Problem with Basic Top-Down parser

Left recursion

- Grammar is a left recursion if it contains at least one non-terminal that has a derivation that includes itself anywhere along its **leftmost branch**
- Example:

NP → NP PP VP → VP PP S → S *and* S	NP → Det Nominal Det → NP *'s*

- Recursion
  - S => [NP VP] => [[NP PP] VP] => [[[NP PP] PP] VP] => …

- Dealing with left-recursion
  - Managing the depth of the search during parsing
  - Rewriting the grammar

$A \rightarrow A\beta \mid \alpha$ ➡ $A \rightarrow \alpha A'$, $A' \rightarrow \beta A' \mid \varepsilon$

Problem with Basic Top-Down parser (cont.)

Ambiguity

- Structural ambiguity
  - Occurs when grammar assigns more than one possible parse to a sentence
- Example of ambiguity

  $[_S\ [_{NP}\ I]\ [_{VP}\ shot\ [_{NP}\ [_{NP}\ an\ elephant]\ [_{PP}\ in\ [_{NP}\ my\ pajamas]]]]]$

  $[_S\ [_{NP}\ I]\ [_{VP}\ shot\ [_{NP}\ an\ elephant]\ [_{PP}\ in\ [_{NP}\ my\ pajamas]]]]$

  [old [man and woman]

  [old man] and [woman]
- Disambiguation algorithm generally require both statistical and semantic knowledge
  - If do not have disambiguation mechanism, parser must return all possible parse trees
  - Modify basic top-down parser to return all possible parses
    - May lead to potentially exponential number of parses that are possible

Probabilistic Context Free Grammar

- PCFG
  - Simplest probabilistic model for recursive embedding
  - Most natural probabilistic model for tree structures
  - A CFG with probability added to the rules, indicating how likely different rewritings are.
- PCFG are one of many ways of building probabilistic models of syntactic structure

Elements of PCFG

- PCFG consists of:
  - A set of terminals; $\{w^k\}$, k = 1, …, V
  - A set of non-terminals: $\{N^i\}$, I = 1, …, n
  - A designated start symbol: N^1
  - A set of rules: $N^i \rightarrow \zeta^j$
  - A corresponding set of probabilities of rules, such that:

$$\forall i \sum_j P(N^i \rightarrow \zeta^j) = 1$$

Assumption of the model

- Place invariance
  - The probability of a subtree does not depend on where in the string the words it dominates are
- Context-free
  - The probability of a subtree does not depend on words not dominates by the subtree
- Ancestor-free
  - The probability of a subtree does not depend on nodes in the derivation outside the subtree

Sentence probability

$$P(w_{1m}) = \sum_{t} P(w_{1m}, t) \quad = \sum_{t:yield(t)=w_{1m}} P(t)$$

- where *t* is a parse tree of the sentence

- We can justify the calculation of the probability of a tree in terms of just multiplying probabilities attached to rules.

Example

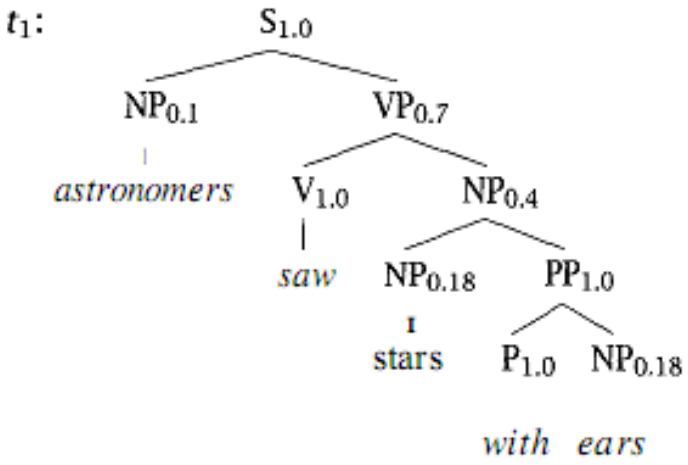
- Grammar example

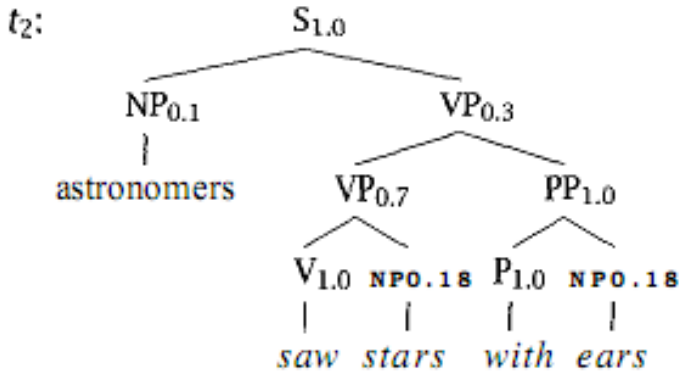
S	→	NP VP	1.0	NP	→	NP PP	0.4
PP	→	P NP	1.0	NP	→	*astronomers*	0.1
VP	→	V NP	0.7	NP	→	*ears*	0.18
VP	→	VP PP	0.3	NP	→	*saw*	0.04
P	→	*with*	1.0	NP	→	*stars*	0.18
V	→	*saw*	1.0	NP	→	*telescopes*	0.1

Example

S	→ NP VP	*1.0*	NP	→ NP PP	*0.4*
PP	→ P NP	*1.0*	NP	→ *astronomers*	*0.1*
VP	→ V NP	*0.7*	NP	→ *ears*	*0.18*
VP	→ VP PP	*0.3*	NP	→ *saw*	*0.04*
P	→ *with*	*1.0*	NP	→ *stars*	*0.18*
V	→ *saw*	*1.0*	NP	→ *telescopes*	*0.1*

- "Astronomers saw stars with ears."





- $P(t_1) = 1.0 * 0.1 * 0.7 * 1.0 * 0.4 * 0.18 * 1.0 * 1.0 * 0.18 = 0.0009072$
- $P(t_2) = 1.0 * 0.1 * 0.3 * 0.7 * 1.0 * 0.18 * 1.0 * 1.0 * 0.18 = 0.0006804$
- $P(w_{1,5}) = P(t_1) + P(t_2) = 0.0015876$

Some Features of PCFG

- Where there are many structurally different parses, PCFG gives some idea of the plausibility of different parses.
  - Not a very good one, since its probability estimates are based purely on structural factors, not factor in lexical co-occurrence.
- PCFG can be learned from positive data alone.
- Robustness
  - Grammatical mistakes and errors in real texts can be avoided to some extent with a PCFG by ruling out nothing in grammar, but by just giving implausible sentences a low probability
- PCFG does not utilize local lexical context
- PCFG are not good models by themselves, but we could hope to combine the strengths of a PCFG and a trigram model

Some Features of PCFG (cont.)

- PCFG has certain biases
  - All else being equal, the probability of a smaller tree is greater than a larger tree
  - PCFG gives too much probability mass to very short sentences.
  - All else being equal, non-terminals with a small number of expansions will be favored over non-terminals with many expansions, since the individual rewritings will have much higher probability

Problems with CFG

- Let's look at the following sentences
  - "A building eats cake."
  - "Mary reads an apple."
  - "I is a cat"
  - "Tom are a dogs.
  - "I went to school tomorrow."
  - "Me was talking to she."
  - "I live a house."
  - "I buy to school."

Grammatical correct, can be successfully parsed with basic CFG

Agreement and Subcategorization

- One consideration for generating grammatical sentences
  - Agreement
    - Formal agreement: the agreement based on grammatical categories
    - Word changes form depending on the other words to which it relates
    - Usually involves making the properties of some grammatical category (such as gender or person) "agree" between varied words or parts of the sentence
  - Subcategorization
    - Ability/necessity for lexical items (usually verbs) to require/allow the presence and types of the syntactic arguments with which they co-occur
- Need to embedded these constraints to grammar rules.

Constraint-based representation

- Why need fine-grained way of representing constraints on grammatical categories?
  - Grammatical phenomena: agreement, subcategorization, etc.
- In Computational linguistics
  - Idea: Object can have complex sets of properties
  - Information in these properties is represented by **constraints**
  - Models are often called **constraint-based formalisms**
- Constraint-based representation
  - Explaining the behavior of larger structures by the combined action of smaller primitives
- Constraint-based representation scheme allow us to represent fine-grained information about…
  - Number and person
  - Agreement
  - Subcategorization
  - Semantic categories, such as mass/count

Agreement

- Agreement
  - Verb and its subject
    - Third-person singular subject VS other kinds of subjects
      - I have a car. VS She has a car.
    - Sentences which the subject does not agree with the verb are ungrammatical
  - Noun's case
    - Nominative: I, he, she, they, we
    - Accusative: me, him, her, them, us
    - Possessive pronoun: my, his, her, their, our
  - Gender agreement (for language, such as German, French)
    - Gender of noun must agree with the gender of its modifying adjective and determiner
  - etc.

Agreement (cont.)

- For example: basic CFG

Grammar	Lexicon
S → NP VP S → Aux NP VP S → VP NP → Det Nominal NP → Proper-Noun Nominal → Noun Nominal → Noun Nominal Nominal → Nominal PP VP → Verb VP → Verb NP	Det → the | this | a Noun → book | flight | meal | money Verb → book | include | prefer Aux → does Prep → from | to | on Proper-noun → Houston | TWA

- How can we modify the grammar to handle these agreement phenomena?
  - Parameterizing each non-terminal of the grammar with feature structures
  - One way is to use a structure called “Feature structure”

Feature Structure

- Encode properties through “feature structures”
  - Set of feature-value pairs
    - Feature: atomic symbols drawn from some finite set
    - Values: atomic symbols or feature structures
  - Traditionally illustrated by matrix-like diagram, called **attribute-value matrix** or **AVM**

$$\begin{bmatrix} \text{Feature}_1 & \text{Value}_1 \\ \text{Feature}_2 & \text{Value}_2 \\ \dots & \\ \text{Feature}_n & \text{Value}_n \end{bmatrix}$$

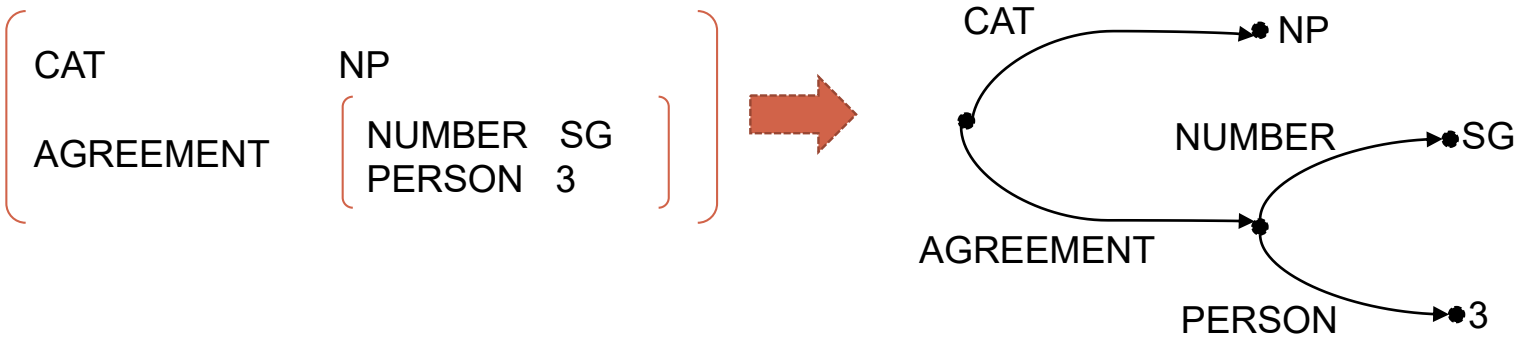
$$\begin{bmatrix} \text{CAT} & \text{NP} \\ \text{NUMBER} & \text{SG} \\ \text{PERSON} & 3 \end{bmatrix}$$

Represent *3sgNP* category

$$\begin{bmatrix} \text{CAT} & \text{NP} \\ \text{AGREEMENT} & \begin{bmatrix} \text{NUMBER} & \text{SG} \\ \text{PERSON} & 3 \end{bmatrix} \end{bmatrix}$$

Feature Structure

- Feature structures as the values
  - Example: subject must agree with their predicates in both number and person
    - Introducing AGREEMENT feature that takes a feature structure consisting of NUMBER and PERSON as its value



Feature Structures in Grammar

- Integrate feature structures and unification operations into the grammar specification
  - Augmenting the CFG rules with attachments that
    - specify feature structures for the constituents of the rules
    - Appropriate unification operations that express constraints
  - Unification
    - **Merging** information content of two structures, **Rejecting** the merger of structures that are incompatible, **Unifying** two feature structures produces a new feature structure
- With such attachments
  - Associate complex feature structures with both lexical items and instances of grammatical categories
  - Enforce compatibility constraints between specified parts of grammatical constructions
  - Guide the composition of feature structures for larger grammatical constituents based on feature structures of their component parts

Unification (1)

- Impose the constraints on agreement and subcategorization by means of unification
- Unification
  - Merging information content of two structures
  - Rejecting the merger of structures that are incompatible
  - Unifying two feature structures produces a new feature structure
    - More specific than, or is identical to, either of the input feature structures
- Unification operator

$$\begin{bmatrix} \text{NUMBER} & \text{SG} \end{bmatrix} \sqcup \begin{bmatrix} \text{NUMBER} & \text{SG} \end{bmatrix} = \begin{bmatrix} \text{NUMBER} & \text{SG} \end{bmatrix}$$

$$\begin{bmatrix} \text{NUMBER} & \text{SG} \end{bmatrix} \sqcup \begin{bmatrix} \text{NUMBER} & \text{PL} \end{bmatrix} = \text{FAIL!!!}$$ ← NUMBER incompatible

$$\begin{bmatrix} \text{NUMBER} & \text{SG} \end{bmatrix} \sqcup \begin{bmatrix} \text{NUMBER} & [] \end{bmatrix} = \begin{bmatrix} \text{NUMBER} & \text{SG} \end{bmatrix}$$ different but compatible

$$\begin{bmatrix} \text{NUMBER} & \text{SG} \end{bmatrix} \sqcup \begin{bmatrix} \text{PERSON} & 3 \end{bmatrix} = \begin{bmatrix} \text{NUMBER} & \text{SG} \\ \text{PERSON} & 3 \end{bmatrix}$$

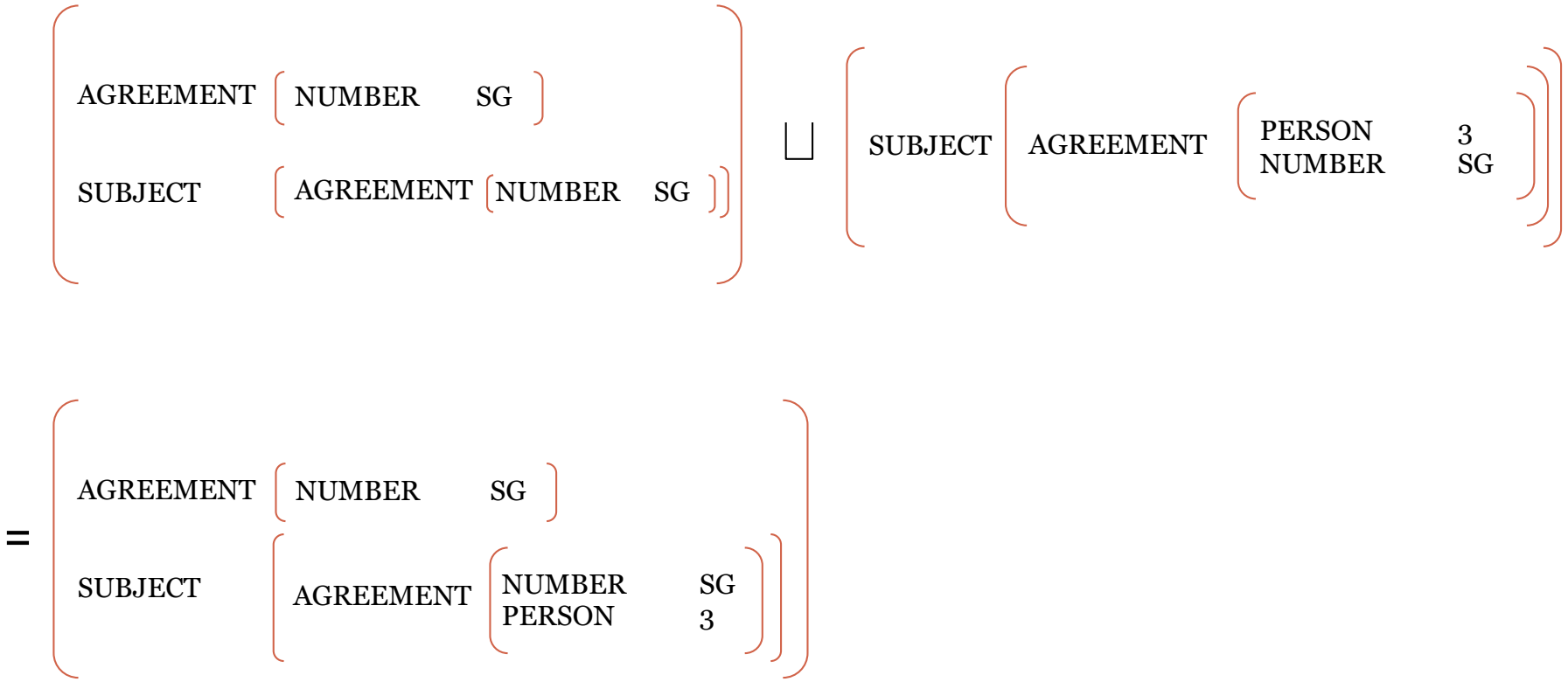
Unification (2)

- Copying capabilities

$$\begin{bmatrix} \text{AGREEMENT} & \boxed{1} \\ \text{SUBJECT} & [\text{AGREEMENT} \quad \boxed{1}] \end{bmatrix} \sqcup \begin{bmatrix} \text{SUBJECT} & [\text{AGREEMENT} & [\text{PERSON} & 3 \\ & & \text{NUMBER} & \text{SG}] \end{bmatrix}$$

$$= \begin{bmatrix} \text{AGREEMENT} & \boxed{1} \\ \text{SUBJECT} & [\text{AGREEMENT} \quad \boxed{1} & [\text{NUMBER} & \text{SG} \\ & & \text{PERSON} & 3] \end{bmatrix}$$

Unification (3)



Grammar Augmentation for Agreement

$\beta_0 \rightarrow \beta_1 \ldots \beta_n$

{set of constraints}

- The specified constraints have one of the following form

  $<\beta_i$ feature path> = Atomic value :: value at the end of the given path must unify with the atomic value

  $<\beta_i$ feature path> = $<\beta_j$ feature path> :: values at the end of the two paths must be unifiable

  Example:

  S → NP VP Correct only if the number of NP is equal to the number of VP

  So …

  S → NP VP

  <*NP* NUMBER> = <*VP* NUMBER>

Grammar Augmentation for Agreement

- Example of agreement features for lexical constituents and rules
  - *Aux* → *do*

    <*Aux* AGREEMENT NUMBER> = PL

    <*Aux* AGREEMENT PERSON> = 3
  - *Determiner* → *this*

    <*Determiner* AGREEMENT NUMBER> = SG
  - *Verb* → *serve*

    <*Verb* AGREEMENT NUMBER> = PL
  - *Verb* → *serves*

    <*Verb* AGREEMENT NUMBER> = SG

    <*Verb* AGREEMENT PERSON> = 3
  - *VP* → *Verb NP*

    <*VP* AGREEMENT> = <*Verb* AGREEMENT>
  - S → NP VP

    <*NP* AGREEMENT> = <*VP* AGREEMENT>

Verb phrase and Subcategorization

- Verb phrase consists of the verb and a number of other constituents
  - NP, PP, combination of NP and PP
  - Clause/Sentence:
    - Embedded sentence as a constituent of verb is called sentence complements
    - You [$_{VP}$ [$_{V}$ said [$_{S}$ there were two patients died yesterday]]]
  - Another VP
    - I [$_{VP}$ want [$_{VP}$ to go to Phuket]]
- Verb phrase can have many possible constituents but not every verb is compatible with every complement/constituent
  - Example: want + NP want + Infinitive
    find + NP ~~find + Infinitive~~
- Subcategorization
  - Traditional grammars: subcategorize verbs into transitive and intransitive verb
  - Modern grammars distinguish as many as 100 subcategories
    - Example: ***find*** subcategorizes for *NP;* ***want*** subcategorizes for either *NP* or a *non-finite VP*

Verb subcategorization

- Possible sets of verb's complements are called the "Subcategorization frame" for the verb.
  - One verb can take different subcategorization frames
  - Example

Frame	Verb	Example
∅	eat, sleep	I want to eat
NP	prefer, find	Find [$_{NP}$ the flight from Bangkok to Phuket]
NP NP	show, give	Show [$_{NP}$ me] [$_{NP}$ airlines with flights from Bangkok]
PP_{from} PP_{to}	fly, travel	I would like to fly [$_{PP}$ from Bangkok] [$_{PP}$ to Phuket]
NP PP_{with}	help, load	Can you help [$_{NP}$ me] [$_{PP}$ with a flight]
VP_{to}	prefer, want, need	I would prefer [$_{VPto}$ to go by Thai Airways]
VP_{brst}	can, would, might	I can [$_{VPbrst}$ go from here]
S	mean	Does this mean [$_{S}$ AA has a hub in Boston]?

Auxiliaries

- Auxiliaries or helping verb (กริยาช่วย)
  - Modal verb: can, could, may, might, must, will, would shall, should
    - Subcategorize for VP whose head V is a bare stem
  - Perfect auxiliary: have, has
    - Subcategorize for VP whose head V is the past participle
  - Progressive auxiliary: be
    - Subcategorize for VP whose head V is the gerundive participle (V+ing)
  - Passive auxiliary: be
    - Subcategorize for VP whose head V is the past participle
- Sentence can have multiple auxiliary verbs
  - But must occur in a particular order: modal < perfect < progressive < passive
    - could have gone, might have been (prevented), will be, has been

Verb subcategorization (cont.)

- Previous examples have shown rather simple subcategorization structures for verbs
- In fact, verbs can subcategorize for quite complex subcategorization frame.
- Verb express subcategorization constraints on subjects as well as complements.
- These subcategorization frame can be composed of many different phrasal types.

- How to come up with a list of subcategorization frames for verbs
  - Need to have a list of possible phrase types that can make up these frames
  - Example: FrameNet
    - https://framenet.icsi.berkeley.edu/fndrupal/
    - Example: https://framenet2.icsi.berkeley.edu/fnReports/data/lu/lu4344.xml?mode=lexentry

Subcategorization in other POS

- Subcategorization is originally designed for verbs
- Other kinds of words exhibit the same behavior
  - while VS during
    - Keep your seatbelt fastened while we are taking off
    - Keep your seatbelt fastened during takeoff.
- Example:
  - It was apparent [$_{Sfin}$ that the kitchen was the only room …]
  - It was apparent [$_{PP}$ from the way she rested her hand over his]
  - aware [$_{Sfin}$ he may have caused offense]
  - It is unimportant [$_{Swheth}$ whether only a little bit is accepted]
  - the assumption [$_{Sfin}$ that wasteful methods have been employed]
  - the question [$_{Swheth}$ whether the authorities might have decided]

Example of NP types for subcategorization

Noun Phrase Types		
There	Nonreferential there	**There** is still much to learn.
It	Nonreferential it	**It** was evident that my ideas …
NP	Noun phrase	As he was relating **his story**
Prepositional Phrase Types		
PP	Prepositional phrase	leaves Bangkok **in the morning**
PPing	Gerundive PP	censured him **for not having intervened**
PPpart	Particle	turn it **off**
Verb Phrase Types		
VPbrst	Bare stem VP	she could **discuss it**
VPto	To-marked infin. VP	Why do you want **to know**?
VPwh	wh-VP	it is worth considering **how to write**
VPing	Gerundive VP	I would consider **using it**

Small set of potential phrase types

Feature Structure with subcategorization

- Using phrase types in a unification grammar by describing each phrase type using features
- For example:
  - **VPto** which is subcategorized for by *want*

Verb → *want*

<*Verb* HEAD SUBCAT FIRST CAT> = VP

<*Verb* HEAD SUBCAT FIRST FORM> = INFINITIVE

VPto

$$\begin{bmatrix} \text{ORTH} & \text{WANT} \\ \text{CAT} & \text{VERB} \\ \text{HEAD} & \begin{bmatrix} \text{SUBCAT} & \langle \begin{bmatrix} \text{CAT} & \text{NP} \end{bmatrix}, \begin{bmatrix} \text{CAT} & \text{VP} \\ \text{HEAD} & \begin{bmatrix} \text{VFORM} & \text{INFINITIVE} \end{bmatrix} \end{bmatrix} \rangle \end{bmatrix} \end{bmatrix}$$

Head features

- Features for most grammatical categories are copied from one of the children to the parent
  - Head of the phrase: Child that provides the features
  - Head features: the features copied
- Example
  - *VP* → *Verb NP*

    <*VP* AGREEMENT> = <*Verb* AGREEMENT>
  - *NP* → *Det Nominal*

    <*NP* AGREEMENT> = <*Nominal* AGREEMENT>
  - Nominal → Noun

    <*Nominal* AGREEMENT> = <*Noun* AGREEMENT>
- Constituent providing the agreement feature structure up to the parent is the head of the phrase.
  - Verb: head of VP Nominal: head of NP Noun: head of Nominal

Head features (2)

- Agreement feature structure is a head feature
  - Rewrite the rules: Placing the agreement feature structure under a HEAD feature, then copying that feature upward
- Example:
  - *VP* → *Verb NP*

    <*VP* HEAD> = <*Verb* HEAD>
  - *NP* → *Det Nominal*

    <*NP* HEAD> = <*Nominal* HEAD>
  - *Nominal* → *Noun*

    <*Nominal* HEAD> = <*Noun* HEAD>
  - *Noun* → *flights*

    <*Noun* HEAD AGREEMENT NUMBER> = PL
  - *Verb* → *serves*

    <*Verb* HEAD AGREEMENT NUMBER> = SG

    <*Verb* HEAD AGREEMENT PERSON> = 3

Subcategorization (cont.)

- Example …

  Verb → serves *(serves breakfast)*

  <*Verb* HEAD AGREEMENT NUMBER> = SG

  <*Verb* HEAD SUBCAT FIRST CAT> = NP

  <*Verb* HEAD SUBCAT SECOND> = END

  Verb → leaves (leaves Bangkok in the morning)

  <*Verb* HEAD AGREEMENT NUMBER> = SG

  <*Verb* HEAD SUBCAT FIRST CAT> = NP

  <*Verb* HEAD SUBCAT SECOND CAT> = PP

  <*Verb* HEAD SUBCAT THIRD> = END

Expect 1 NP argument as its object

- For VP rule …

  VP → verb NP

  <*Verb* HEAD SUBCAT FIRST CAT> = <NP CAT>

  <*Verb* HEAD SUBCAT SECOND> = END

  <*VP* HEAD> = <*Verb* HEAD>

FIRST_CAT of verb's SUBCAT must match the category of constituent immediately following verb

Linguistic Resources

- Treebank
- FrameNet
- PropBank
- VerbOcean

Dependency Tree

- He will bring a book to me tomorrow.

  nsubj(bring-3, He-1)

  aux(bring-3, will-2)

  root(ROOT-0, bring-3)

  det(book-5, a-4)

  dobj(bring-3, book-5)

  case(me-7, to-6)

  nmod(bring-3, me-7)

  nmod:tmod(bring-3, tomorrow-8)