Formal Grammars of English

Dr. Demetrios Glinos
University of Central Florida

CAP6640 – Computer Understanding of Natural Language

Today

- Syntax
- Context-Free Grammars
- Grammar Rules for English
- Treebanks
- Grammar Equivalence and Normal Form
- Lexicalized Grammars

Constituents

- We do not understand words in isolation
- We group them into meaningful units called **constituents**
- We relate the units in *hierarchical* ways

- Consider the sentence

  The man with the umbrella walked into the store .

  Meaningful units: *the man with the umbrella*
walked into the store

- Constituents may be further decomposed: the + man
with + the umbrella
walked + into the store
etc.

Syntax and Grammar

- **Syntax**
  - refers to the way words are arranged together
- **Grammar**
  - mechanism for determining valid and invalid arrangements (constituents)
    - Valid

      The man with the umbrella walked into the store .

    - Invalid

      into umbrella man the with the walked store the .

- We use **formal language theory** to study grammars

Formal Language Theory

- Attributable to linguist Noam Chomsky (1956)
  - **Language**
    - a collection of sentences of finite length all constructed from a finite alphabet of symbols
  - **Grammar**
    - a mechanism that enumerates the sentences of a language
    - a grammar of language L can be regarded as a function whose range is exactly L
  - **Formal grammar**
    - a precisely defined grammar
  - **Generative grammar**
    - a formal grammar that can "generate" natural language expressions

Levels of Representation

- A sentence in a language has two levels of representation
  - **Deep structure**
    - a direct representation of the semantics underlying the sentence
  - **Surface structure**
    - the syntactical representation
- Deep structures are mapped onto surface structures via ***transformations***
- Example:

  Peter put the book on the table
  The book was put on the table by Peter
  } same semantics

Formal Grammar

A **formal grammar** is a quad-tuple $\boldsymbol{G} = (\boldsymbol{N}, \boldsymbol{\Sigma}, \boldsymbol{P}, \boldsymbol{S})$, where

$\boldsymbol{N}$ is a finite set of non-terminals

for which some production rule can be applied

$\boldsymbol{\Sigma}$ is a finite set of terminal symbols

disjoint from N

for which no production rule can be applied

$\boldsymbol{P}$ is a finite set of production rules of the form

$$\boldsymbol{w} \in (\boldsymbol{N} \cup \boldsymbol{\Sigma})^* \rightarrow \boldsymbol{w} \in (\boldsymbol{N} \cup \boldsymbol{\Sigma})^*$$

where $\boldsymbol{w}$ ("word") is a valid sequence of symbols in the language

$\boldsymbol{S} \in N$ is the start symbol

Formal Grammar

A **formal grammar** is a quad-tuple $G = (N, \Sigma, P, S)$, where

N is a finite set of non-terminals

Σ is a finite set of terminals and is disjoint from N

P is a finite set of production rules of the form

$$w \in (N \cup \Sigma)^* \rightarrow w \in (N \cup \Sigma)^*$$

$S \in N$ is the start symbol

A formal language

- provides an *axiom schema* for generating a (usually infinite) set of finite-length sequences of symbols
- sequences may be constructed by successive applications of the production rules
- a rule may be applied by replacing an occurrence of the symbols on the left-hand side with those that appear on its right-hand side
- a sequence of rule applications is called a *derivation*
- the language so defined is the set of all "words" (sequences of terminals) that can be reached from the start symbol

Notation and Example

- Notation
  - Upper case – nonterminals
  - Lower case – terminals
  - S – Start symbol

- Example

  Let L have terminals { *a, b* }, nonterminals { *S, A, B* }, start symbol *S*, and production rules:

  $S \rightarrow AB$

  $S \rightarrow \varepsilon$ (where ε is the empty string)

  $A \rightarrow aS$

  $B \rightarrow b$

  This grammar defines the language of all words of the form $a^n b^n$

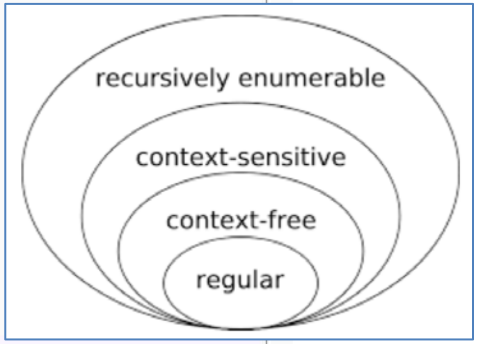
The Chomsky Hierarchy

Each grammar type strictly includes all grammars below it in the hierarchy

Grammar	Languages	Automaton	Production rules (constraints)*
Type-0	Recursively enumerable	Turing machine	$\alpha \rightarrow \beta$ (no restrictions)
Type-1	Context-sensitive	Linear-bounded non-deterministic Turing machine	$\alpha A \beta \rightarrow \alpha \gamma \beta$
Type-2	Context-free	Non-deterministic pushdown automaton	$A \rightarrow \gamma$
Type-3	Regular	Finite state automaton	$A \rightarrow a$ and $A \rightarrow aB$

* Meaning of symbols:

a = terminal
α = terminal, non-terminal, or empty
β = terminal, non-terminal, or empty
γ = terminal or non-terminal
A = non-terminal
B = non-terminal



source: Wikipedia

Today

- Syntax
- Context-Free Grammars
- Grammar Rules for English
- Treebanks
- Grammar Equivalence and Normal Form
- Lexicalized Grammars

Context-Free Grammars

- Also known as "phrase-structure grammars"
- Basis of many formal models of the syntax of natural languages
  - also for computer progrmming languages
- Powerful enough to express sophisticated relations among words in a sentence
- Computationally tractable, supporting efficient algorithms for parsing
- Uses
  - grammar checking
  - semantic interpretation
  - dialogue understanding
  - machine translation

Example: ATIS L_0

- Air Traffic Information System (ATIS) (1990)
  - one of the earliest spoken language systems, for booking airline reservations

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*

Figure 11.2 The lexicon for $\mathscr{L}_0$.

source: J&M (3d Ed. draft)

Example: ATIS L_0

Grammar Rules			Examples
S	→	*NP VP*	I + want a morning flight
NP	→	*Pronoun*	I
	|	*Proper-Noun*	Los Angeles
	|	*Det Nominal*	a + flight
Nominal	→	*Nominal Noun*	morning + flight
	|	*Noun*	flights
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

Figure 11.3 The grammar for $\mathscr{L}_0$, with example phrases for each rule.

source: J&M (3d Ed. draft)

Example: ATIS L_0

Grammar Rules			Examples
S	→	*NP VP*	I + want a morning flight
NP	→	*Pronoun*	I
	|	*Proper-Noun*	Los Angeles
	|	*Det Nominal*	a + flight
Nominal	→	*Nominal Noun*	morning + flight
	|	*Noun*	flights
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

Figure 11.3 The grammar for $\mathscr{L}_0$, with example phrases for each rule.

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*

Figure 11.2 The lexicon for $\mathscr{L}_0$.

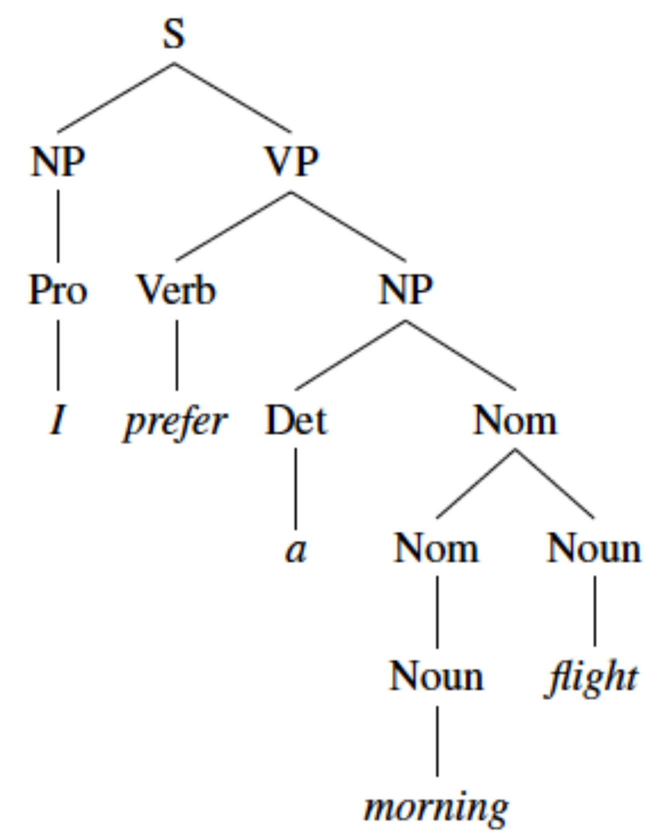


Figure 11.4 The parse tree for “I prefer a morning flight” according to grammar $\mathscr{L}_0$.

source: J&M (3d Ed. draft)

Syntactic Parsing

Definition:

If $A \rightarrow \beta$ is a production of R and α and γ are any strings in the set $(\Sigma \cup N)^*$, then we say that $\alpha A \gamma$ ***directly derives*** $\alpha\beta\gamma$, and we write $\alpha A \gamma \Rightarrow \alpha\beta\gamma$

Definition:

Let $\alpha_1, \alpha_2, \ldots, \alpha_m$ be strings in $(\Sigma \cup N)^*, m \geq 1$, such that

$$\alpha_1 \Rightarrow \alpha_2, \alpha_2 \Rightarrow \alpha_3, \ldots, \alpha_{m-1} \Rightarrow \alpha_m$$

then we say that α_1 ***derives*** α_m, and we write $\alpha_1 \Rightarrow^* \alpha_m$

Definition:

The language generated by grammar G is $\mathcal{L}_G = \{w | w \text{ is in } \Sigma^* \text{ and } S \Rightarrow^* w\}$

Syntactic parsing

the problem of mapping from a string of words to its parse tree is called

Today

- Syntax
- Context-Free Grammars
- Grammar Rules for English
- Treebanks
- Grammar Equivalence and Normal Form
- Lexicalized Grammars

Sentence-Level Constructions

- English has 4 major sentence-level constructions

1. Declarative

    - basic form is S ⟶ NP VP, where we typically call the NP the "subject"

    - examples

        - I want a flight from Ontario to Chicago
        - The flight should depart at eleven a.m. tomorrow
        - The return flight should leave at around seven p.m.

Sentence-Level Constructions

2. Imperative

- basic form is S ⟶ VP ; typically has no subject; used for commands, etc.
- examples
  - Show the lowest fares
  - Give me Sunday's flights from New York City to Las Vegas
  - List all flights between five and seven p.m.

Sentence-Level Constructions

3. Yes-No question

- basic form is S ⟶ Aux NP VP ; begins with auxiliary verb, then subject NP, then VP

- examples
  - Do any of these flights have stops?
  - Does American's flight eighteen twenty five serve dinner?
  - Can you give me the same information for United?

Sentence-Level Constructions

4. Wh-questions
   - contains a "wh-phrase"
     - includes a "wh-word": who, whose, when, where, what, which, how, why
   - wh-subject-question (S ⟶ Wh-NP VP)
     - identical to declarative structure, except first NP contains a wh-word
     - examples:
       - What airlines fly from Burbank to Denver?
       - Whose flights serve breakfast?
   - wh-non-subject-question (S ⟶ Wh-NP Aux NP VP)
     - wh-phrase is not the subject; auxiliary appears before the subject NP
     - example
       - What flights do you have from Burbank to Tacoma Washington?

Note the long-distance dependency: the leading Wh-NP is the object of the VP

Clauses and Sentences

- Sentence constructions ("S ⟶ " rules)
  - represent a "complete thought"
  - correspond to the lingistic notion of a ***clause***
  - can be embedded within a larger sentence
  - e.g., "I said I prefer a morning flight."

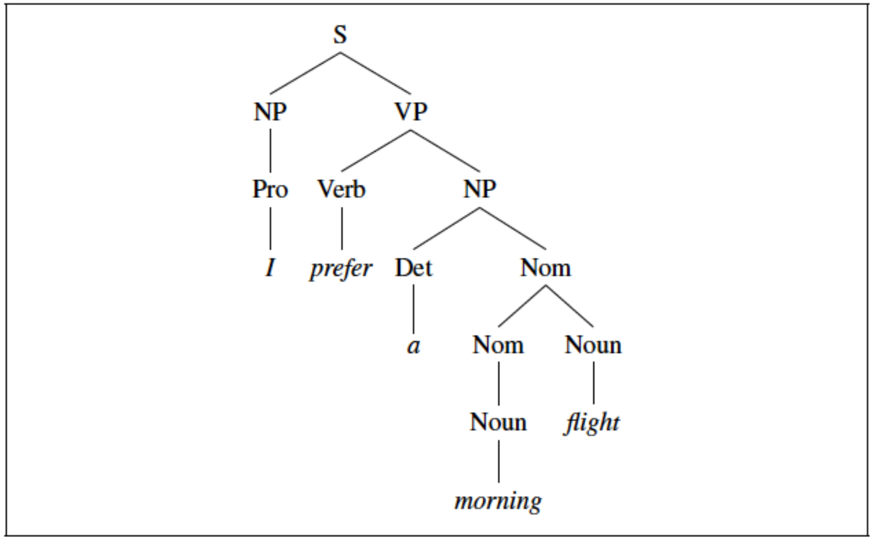


Figure 11.4 The parse tree for "I prefer a morning flight" according to grammar $\mathscr{L}_0$.

- In other words
  - S is a node in a parse tree below which the main verb of the S has all of its arguments

The Noun Phrase

- L_0 contains these NP productions:

NP	→	*Pronoun*	I
	|	*Proper-Noun*	Los Angeles
	|	*Det Nominal*	a + flight

- The complexity is in the "Det Nominal" rule
- Determiner
  - can be simple: a, an, the, those, this, some, any, ...
    - e.g., those flights, a stop, the plane
  - can be filled by more complex expressions
    - Det ⟶ NP 's
    - e.g., United's pilot's union (*which also illustrates NP recursion*)
  - optional when the noun is plural or mass noun
    - e.g., Show me flights from New York to Denver on weekdays

The Noun Phrase

Nominal	$\rightarrow$	*Nominal Noun*	morning + flight
	$\mid$	*Noun*	flights

- Nominal
  - follows the determiner
  - contains any pre- and post-head noun modifiers
    - before the head noun ("postdeterminers")
      - cardinal numbers, ordinal numbers, quantifiers, and adjectives
        - e.g., the *first* flight, *many* fares, the *longest* layover
      - adjectives can be grouped into an adjective phrase (AP), which can have an adverb before the adjective
        - e.g., the *least expensive* fare
    - after the head noun ("postmodifiers")

multiple PPs

      - prepositional phrases, e.g., all flights *from Cleveland to Newark*
      - non-finite clauses, e.g., any flights *arriving after eleven a.m.*
      - relative clauses e.g., a flight *that serves breakfast*

The Nominal (cont'd)

- Non-finite postmodifiers
  - gerundive
    - postmodifier consists of a VP that begins with the gerundive (-*ing*)
    - examples:
      - any of those *leaving on Thursday*
      - flights *arriving within the next two hours*
  - infinitive
    - postmodifier consists of a VP that begins with the infinitive (*to+*)
    - e.g., the last flight *to arrive in Boston*
  - -ed form
    - postmodifier consists of a VP that begins with the past participle (-*ed*)
    - e.g., the aircraft *used by this flight*

The Nominal (cont'd)

- Postnominal relative clause
  - also called "restrictive relative clause"
  - typically begins with a relative pronoun (e.g., *that* and *who*),
    - can serve as the subject of the embedded verb
      - examples:
        - a flight *that serves breakfast*
        - flights *that leave in the morning*
    - can serve as the object of the embedded verb
      - e.g., the earliest American Airlines flight *that I can get*

- Predeterminers
  - word classes that modify and appear before NPs
  - typically involve number or amount; "all" is a common example
  - examples: all the flights; all flights; all non-stop flights

Example: Complex NP

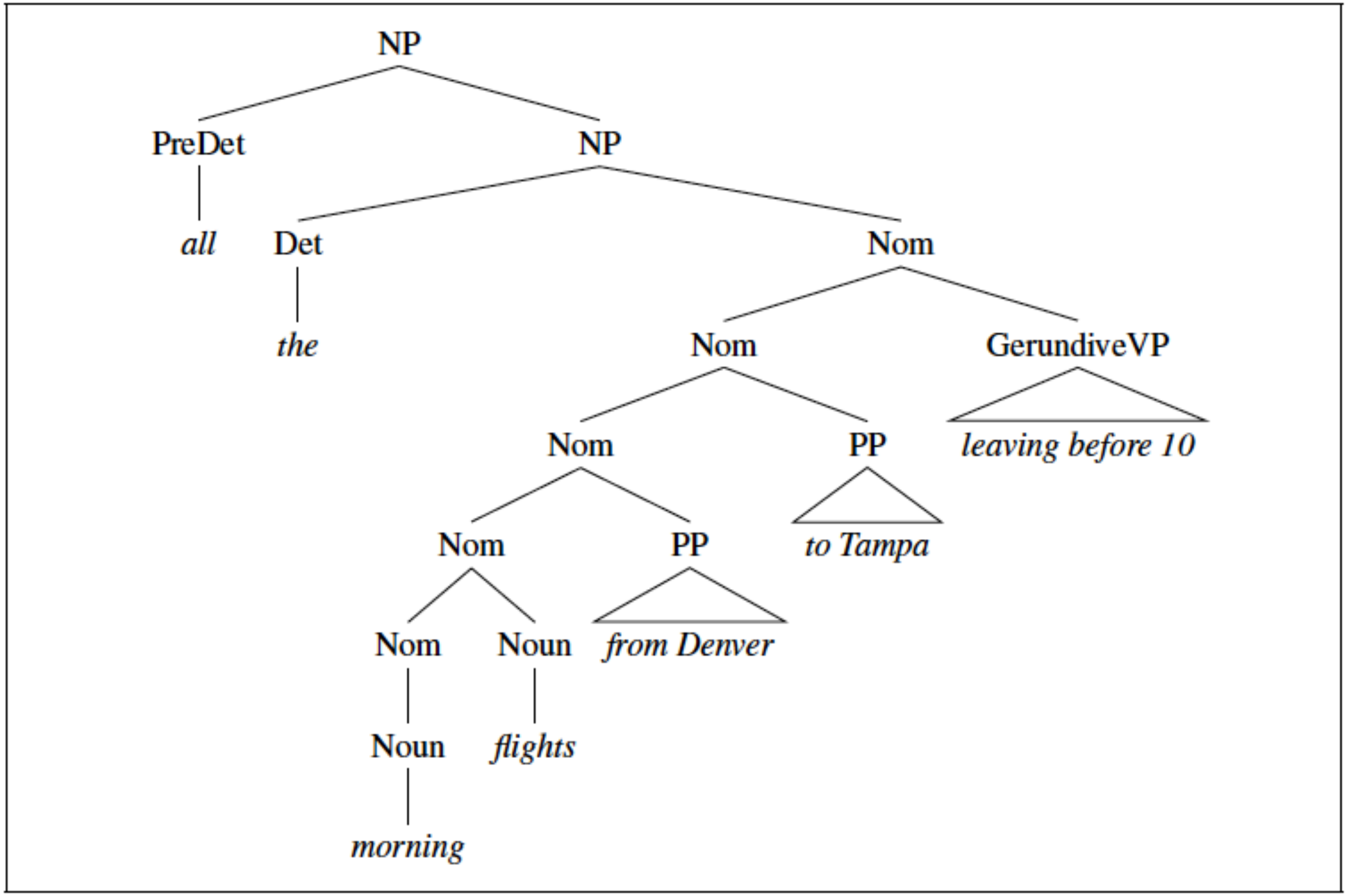


Figure 11.5 A parse tree for "all the morning flights from Denver to Tampa leaving before 10".

source: J&M (3d Ed. draft)

The Verb Phrase

- L_0 contains these VP productions:

VP	$\rightarrow$	$Verb$	do
	$\mid$	$Verb\ NP$	want + a flight
	$\mid$	$Verb\ NP\ PP$	leave + Boston + in the morning
	$\mid$	$Verb\ PP$	leaving + on Thursday

- sentential complements (VP ⟶ Verb S)
  - entire embedded sentences that can also follow VPs
  - e.g., You said *you had a two hundred sixty six dollar fare*

- VPs can also be followed by other VPs
  - common for verbs like *want, would like, try, intend, need*, etc.
  - e.g., I want *to fly from Milwaukee to Orlando*

Verb Subcategorization Frames

Frame	Verb	Example
$\emptyset$	eat, sleep	I ate
NP	prefer, find, leave	Find [$_{NP}$ the flight from Pittsburgh to Boston]
NP NP	show, give	Show [$_{NP}$ me] [$_{NP}$ airlines with flights from Pittsburgh]
$PP_{\text{from}}\ PP_{\text{to}}$	fly, travel	I would like to fly [$_{PP}$ from Boston] [$_{PP}$ to Philadelphia]
$NP\ PP_{\text{with}}$	help, load	Can you help [$_{NP}$ me] [$_{PP}$ with a flight]
VPto	prefer, want, need	I would prefer [$_{VPto}$ to go by United airlines]
VPbrst	can, would, might	I can [$_{VPbrst}$ go from Boston]
S	mean	Does this mean [$_{S}$ AA has a hub in Boston]

Figure 11.6 Subcategorization frames for a set of example verbs.

source: J&M (3d Ed. draft)

- Notes:
  - traditional "transitive" and "intransitive" distinctions have been replaced by finer-grained subcategorizations, as above
  - Not every verb is compatible with every VP subcategorization
    - e.g., "want" can have NP or VP_{to} complements
    - e.g., "find" cannot take VP_{to} complement

Coordination

- Coordinating conjunctions
  - and, but, or
  - can join NPs, VPs, and Ss to form larger constructions of the same type

  - example NP coordination
    - please repeat *the flights and costs*
    - I need to know *the aircraft and the flight number*

  - example VP coordination
    - What flights do you have *leaving Denver and arriving in San Francisco?*

  - example S coordination
    - I am interested in a flight from Dallas to Washington *and* I'm also interested in going to Baltimore

Today

- Syntax
- Context-Free Grammars
- Grammar Rules for English
- Treebanks
- Grammar Equivalence and Normal Form
- Lexicalized Grammars

Treebank

- A syntactically annotated corpus
- Made possible by a sufficiently robust CFG that ensures that every sentence will have at least one parse
- Typically generated using automated parsers, with human linguists to hand-correct the parses
- Treebank projects
    - Penn Treebank
        - corpora include Brown, Switchboard, ATIS, Wall Street Journal for English, plus various Chinese and Arabic corpora
    - Prague Dependency Treebank for Czech
    - Negra Treebank for German
    - Susanne Treebank for English

Penn Treebank Example: WSJ

```
( (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) ))
    (. .) ))
```

Figure 11.9 A sentence from the *Wall Street Journal* portion of the LDC Penn Treebank. Note the use of the empty -NONE- nodes.

source: J&M (3d Ed. draft)

Treebanks as Grammars

- The annotated sentences in a treebank implicitly constitute a grammar
- We can extract the grammar's production rules from these sentences
- But such extracted grammars are very "flat"
  - very many rules
  - very long rules
  - e.g., 4,500 different rules for expanding VPs
  - example:

    Rule: VP $\longrightarrow$ VBP PP PP PP PP PP ADVP PP

    Which comes from VP in:

    This mostly happens because we *go from football in the fall to lifting in the winter to football again in the spring.*
- As a result, it is common to make modifications to a grammar extracted from a treebank

Heads and Head Finding

- Basic idea: each syntactic constituent can be associated with a lexical "head"
  - e.g., that a N is the head of an NP, a V is the head of a VP
  - a basic notion in linguistics since the early 20th century

  - central to grammar formalisms
    - Head-Driven Phrase Structure Grammar
    - dependency relations in grammars

  - in computational linguistics
    - for probabilistic parsing
    - for dependency parsing

Finding Lexical Heads

- A simple model of lexical heads (Charniak 1997, Collins 1999)
  - augment each CFG rule with the word in the phrase that is grammatically the most important
  - heads are passed up the parse tree
  - each nonterminal is annotated with a single word: its lexical "head"
- Basic approach
  - each CFG rule must be augmented to identify one right-side constituent to be the head daughter

Finding Lexical Heads

- Alternative lexicalization approach
  - Parse the sentence
  - After parsing, use a simple set of handwritten rules to "decorate" each node with the appropriate head

- Example: Collins's rule for determining the head of an NP:

> - If the last word is tagged POS, return last-word.
> - Else search from right to left for the first child which is an NN, NNP, NNPS, NX, POS, or JJR.
> - Else search from left to right for the first child which is an NP.
> - Else search from right to left for the first child which is a $, ADJP, or PRN.
> - Else search from right to left for the first child which is a CD.
> - Else search from right to left for the first child which is a JJ, JJS, RB or QP.
> - Else return the last word

source: J&M (3d Ed. draft)

Collins Lexicalized Parse

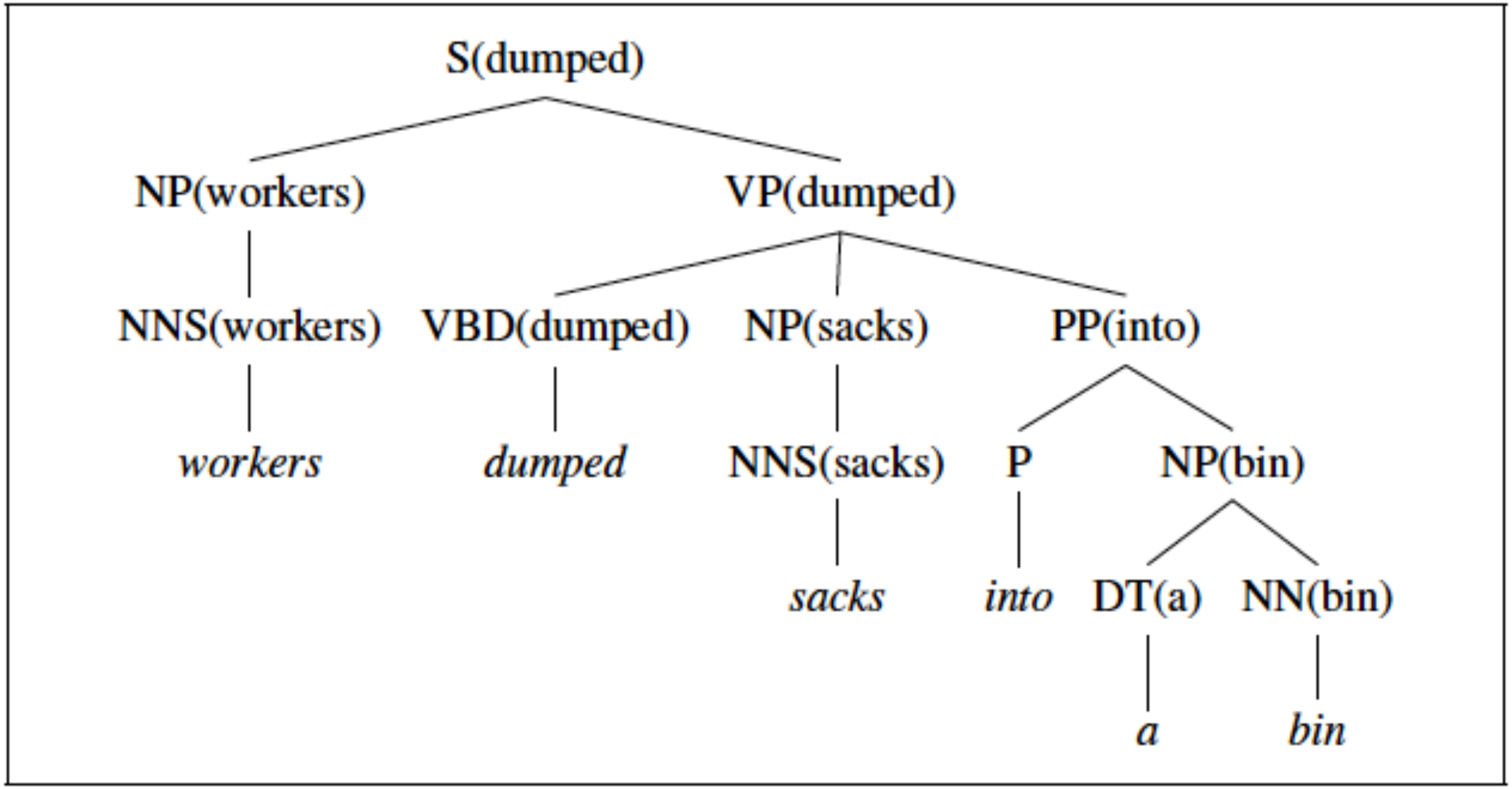


Figure 11.11 A lexicalized tree from Collins (1999).

source: J&M (3d Ed. draft)

Today

- Syntax
- Context-Free Grammars
- Grammar Rules for English
- Treebanks
- Grammar Equivalence and Normal Form
- Lexicalized Grammars

Grammar Equivalence

- Recall: a formal language is a (possibly infinite) set of strings
- It is possible that two different CFGs generate the same language
- **weak equivalence**
  - 2 grammars generate the same set of strings
- **strong equivalence**
  - 2 grammars generate the same set of strings
  - assign the same phrase structure to each sentence

Normal Form for Grammars

- **Chomsky Normal Form (CNF)**
  - productions are of form $A \rightarrow B\ C$
    $A \rightarrow a$
  - no production produces the empty string (i.e., do not have $A \rightarrow \varepsilon$)
- Result is binary branching (useful for parsing)
- Any CFG can be converted into a weakly equivalent CNF

Example: $A \rightarrow B\ C\ D$

can be converted to $A \rightarrow B\ X$
$X \rightarrow C\ D$

Today

- Syntax
- Context-Free Grammars
- Grammar Rules for English
- Treebanks
- Grammar Equivalence and Normal Form
- Lexicalized Grammars

Lexicalized Grammars

- CFGs have difficulty handling
  - agreement (person, number, tense)
  - subcategorization
  - long-distance dependencies

- "Lexicalized" grammar formalisms attempt to make better use of the lexicon
  - Lexical-Function Grammar (LFG)
  - Head-Driven Phrase Structure Grammar (HPSG)
  - Tree-Adjoining Grammar (TAG)
  - Combinatory Categorial Grammar (CCG)

Combinatory Categorial Grammar

- Major elements
  - a set of *categories*
  - a *lexicon* that associates words with categories
  - a set of *rules* for how categories combine in context

- Categories
  - either atomic elements or single-argument functions that return a category as a value given a desired category as input

Given set of categories $\mathcal{C}$ and a set of atomic elements $\mathcal{A}$, where $\mathcal{A} \subseteq \mathcal{C}$, we define

$$(X\,/\,Y),\ (X\setminus Y) \in \mathcal{C},\ \text{ if } X, Y \in \mathcal{C}$$

where

$(X\,/\,Y)$ is the function that seeks a constituent Y to right of X, and returns X

$(X \setminus Y)$ is the function that seeks a constituent Y to left of X, and returns X

Categorial Grammar: Lexicon

- The lexicon assigns categories to words
  - can assign to atomic or functional categories
  - a word can be assigned to multiple categories

- Example lexical entries

flight:	N
Miami:	NP
cancel:	(S \ NP) / NP

seek NP to right

returns a function

- The returned function can combine with NP on left to return an S as result
- This formalism captures verb subcategorization information for the transitive verb "cancel"

Categorial Grammar: Rules

- Categorial grammar rules specify how functions and their arguments combine

- Only 2 rule templates for all categorial grammars

$X/Y \quad Y \quad \Rightarrow \quad X$ ← forward function application

$Y \quad X\backslash Y \quad \Rightarrow \quad X$ ← backward function application

- Example

Given: *United* and *Miami* are both simple NPs

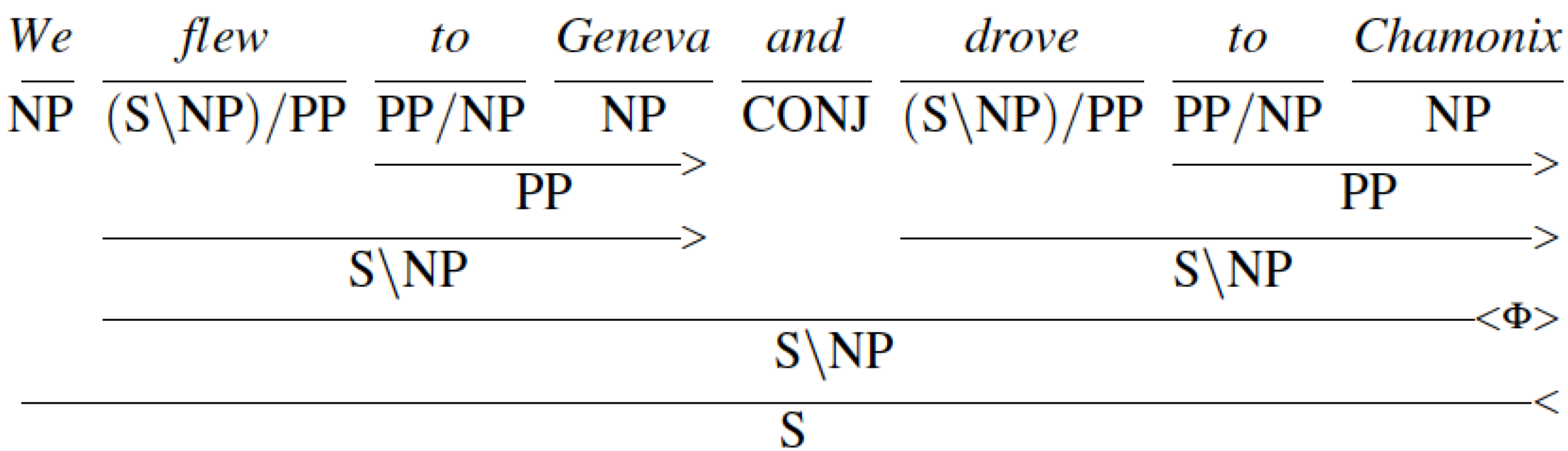
serves is transitive with category (S \ NP) / NP

We analyze: *United* *serves* *Miami*

$$\underbrace{NP \quad \underbrace{(S \backslash NP) / NP \quad NP}_{S \backslash NP}}_{S}$$

Categorial Grammar: Conjunctions

- Conjunctions can be handled by the simple rule: X CONJ X $\Rightarrow$ X



source: J&M (3d Ed. draft), Chap. 11

Combinatory Categorial Grammar

- Basic categorial grammar no more expressive than traditional CFG
  - grammatical facts just basically pushed down into the lexicon

- CCG is more expressive by including operations on functions

  - Composition operations

$$X / Y \quad Y / Z \Rightarrow X / Z$$
$$Y \backslash Z \quad X \backslash Y \Rightarrow X \backslash Z$$

  - Type raising operations
    - elevates simple categories to the status of functions

$$X \Rightarrow T / (T \backslash X)$$
$$X \Rightarrow T \backslash (T / X)$$

    - where T can be any atomic or functional category already in the grammar

CCG Example: Type raising

- We can use type raising to reinvent an NP as a function in its own right

$$NP \Rightarrow S / (S \backslash NP)$$

- Type-raising alternative for previous example

United	*serves*	*Miami*
NP	(S \ NP) / NP	NP

$\underbrace{\text{NP}}_{>T}$

S / (S \ NP)

$\underbrace{\text{S / (S \ NP) \quad (S \ NP) / NP}}_{>B}$

S / NP

S

Note how this provides a left-to-right, word-by-word derivation that more closely mirrows how humans process language

CCG: Handling long-distance dependencies

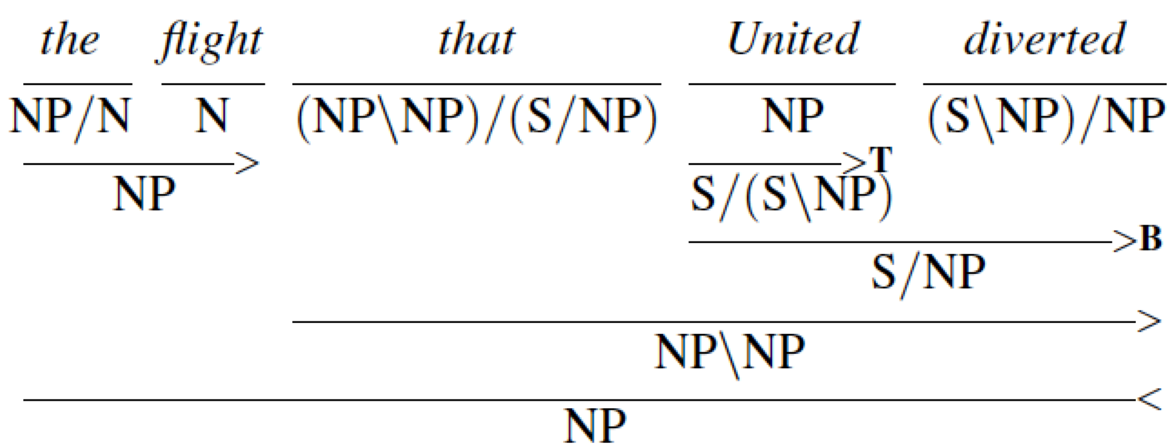
- Consider the relarive clause: "the flight that United diverted"

  divert (S \ NP) / NP

  transitive verb: subject NP to left + direct object NP to right

  but here, the direct object NP ("the flight") has been moved to the beginning of the clause

- derivation:



source: J&M (3d Ed. draft), Chap. 11

Summary of Formal Grammars

- Formal grammars can represent much of the richness of formal languages
- Traditional CFGs provide a baseline for many language features
- Lexicalized grammars can handle additional features
    - e.g., subcategorization, long-distance dependencies