CS 4248
Natural Language Processing

Professor NG Hwee Tou
Department of Computer Science
School of Computing
National University of Singapore
nght@comp.nus.edu.sg

Chapter 12: Formal Grammars of English

- Syntax: Relationships between words
  - Order of words
  - Grouping of words
  - Structure of sentences

Constituency

- Constituent
  - A group of words that behaves as a single unit
- Examples:
  - Noun phrase / noun group (NP)
  - Prepositional phrase (PP)

Noun Phrase

- A sequence of words surrounding at least one noun
- Examples:
  - they
  - a tall man
  - George Bush
  - three flights from California
  - the flight that serves breakfast

Evidence of Constituency

- Appearance in similar syntactic environments (e.g., before a verb)
  - they spoke …
  - a tall man waved …
  - George Bush announced …
  - three flights from California arrive …
  - the flight that serves breakfast costs …

  But not true of the individual words
  - *tall waved
  - *from arrive
  - *serves costs

Evidence of Constituency

- Preposed or postposed constructions
  - On May seventh, I'd like to fly to Sydney
  - I'd like to fly on May seventh to Sydney
  - I'd like to fly to Sydney on May seventh

  But not true of the individual words
  - *On May, I'd like to fly seventh to Sydney
  - *On I'd like to fly May seventh to Sydney
  - *I'd like to fly on May to Sydney seventh

Context-Free Grammar (CFG)

- Also called Constituency Grammar or Phrase-Structure Grammar or Backus-Naur Form (BNF)
- Example of a CFG:

  NP $\rightarrow$ Det Nominal

  Nominal $\rightarrow$ Noun | Noun Nominal

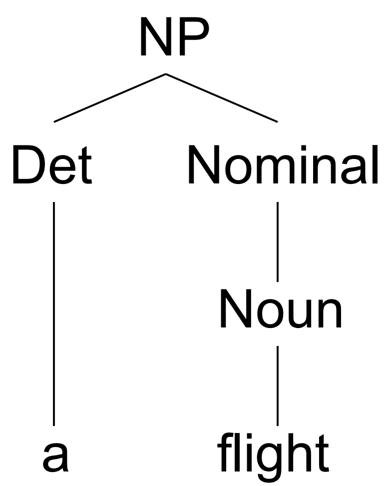
  Det $\rightarrow$ a | the

  Noun $\rightarrow$ flight | trip
- Generator or recognizer

CFG

- Derivation:

  NP $\Rightarrow$ Det Nominal $\Rightarrow$ Det Noun $\Rightarrow$ a Noun $\Rightarrow$ a flight
- Parse tree:

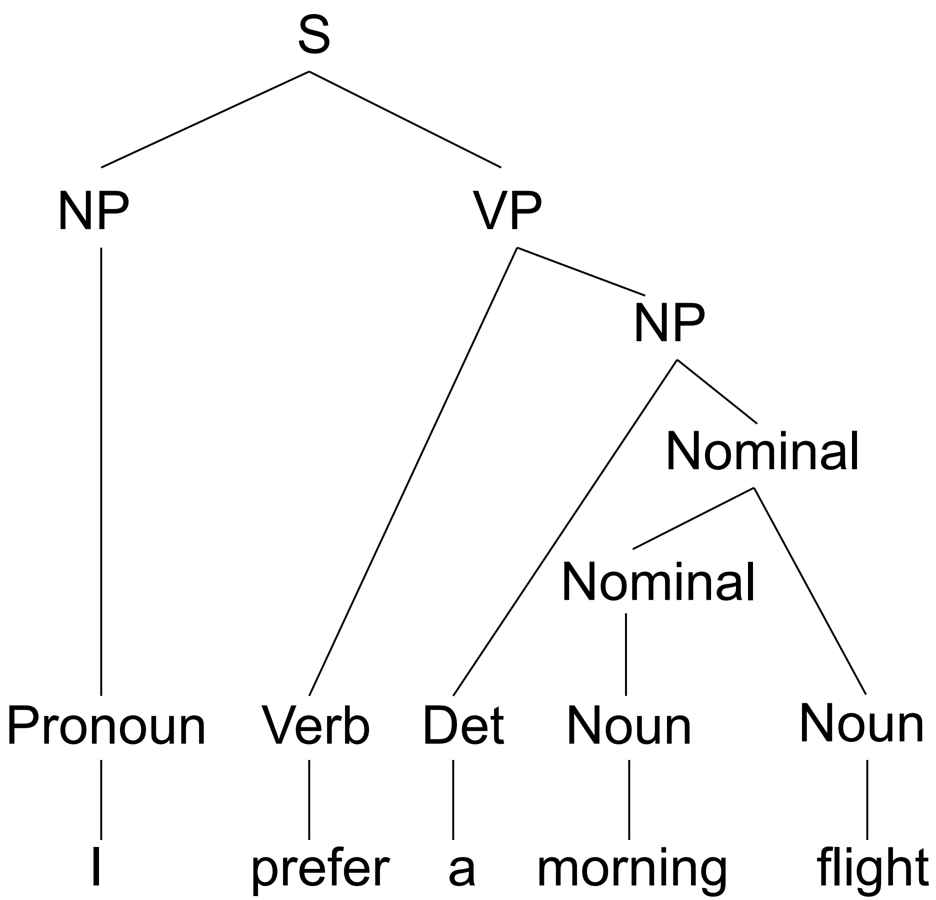


The node NP **dominates** all the nodes Det, Nominal, Noun, a, flight
The node NP **immediately dominates** the nodes Det, Nominal

CFG

$S \rightarrow$ NP VP
NP $\rightarrow$ Pronoun | ProperNoun | Det Nominal
Nominal $\rightarrow$ Nominal Noun | Noun
VP $\rightarrow$ Verb | Verb NP | Verb PP | Verb NP PP
PP $\rightarrow$ Preposition NP
Pronoun $\rightarrow$ I | you | he | she | me | it | …
ProperNoun $\rightarrow$ Alaska | Baltimore | …
Det $\rightarrow$ a | an | the | this | these | that | …
Noun $\rightarrow$ flight | trip | morning | …
Verb $\rightarrow$ is | prefer | like | need | want | fly | …
Preposition $\rightarrow$ from | to | on | near | …

CFG



Constituency
parse tree

CFG

- Bracketed notation

  $[_S[_{NP}[_{Pronoun}$ I]]$[_{VP}[_{Verb}$ prefer]$[_{NP}[_{Det}$ a]$[_{Nominal}$ $[_{Nominal}[_{Noun}$ morning]] $[_{Noun}$ flight]]]]]

- Grammatical (vs. ungrammatical)
  - A string that can (vs. cannot) be derived by a grammar
- Generative grammar (in Linguistics)
  - The use of formal languages to model natural languages
  - The language is defined by the set of possible sentences "generated" by the grammar

Formal Definition of CFG

- $G = (N, \Sigma, P, S)$
  - A set of non-terminal symbols (or variables) N
  - A set of terminal symbols Σ ($N \cap \Sigma = \varnothing$)
  - A set of productions P, each of the form $A \rightarrow \alpha$, where $A \in N$ and $\alpha \in (\Sigma \cup N)^*$
  - A designated start symbol $S \in N$

CFG

- If $A \rightarrow \beta$ is a production of P, and α and γ are any strings in $(\Sigma \cup N)^*$, then $\alpha A \gamma$ *directly derives* $\alpha\beta\gamma$, or $\alpha A \gamma \Rightarrow \alpha\beta\gamma$
- Suppose that $\alpha_1, \alpha_2, \ldots, \alpha_m$ are strings in $(\Sigma \cup N)^*$, $m \geq 1$, and

  $\alpha_1 \Rightarrow \alpha_2$

  $\alpha_2 \Rightarrow \alpha_3$

  …

  $\alpha_{m-1} \Rightarrow \alpha_m$

  then α_1 *derives* α_m or $\alpha_1 \Rightarrow^* \alpha_m$

CFG

- The language generated by a grammar G, denoted L(G), is the set of strings composed of terminal symbols which can be derived from the start symbol S of G

  $L(G) = \{ w \mid w \in \Sigma^* \text{ and } S \Rightarrow^* w \}$
- Syntactic parsing: Mapping from a string of words to its parse tree

CFG

- Strong equivalence: Two grammars G1 and G2 are strongly equivalent if they generate the same set of strings (i.e., L(G1) = L(G2)) **and** they assign the same phrase structure to each string (allowing only for renaming of non-terminal symbols)
- Weak equivalence: Generate the same set of strings but do not assign the same phrase structure to each string

CFG

- Chomsky Normal Form (CNF):
  - The grammar is ε-free
  - Each production of the grammar is either of the form $A \rightarrow B\ C$ or $A \rightarrow a$ (i.e., either 2 non-terminal symbols or 1 terminal symbol on RHS)
- Any CFG can be converted into a weakly equivalent CFG in Chomsky Normal Form

Algorithm to Convert a CFG with No ε-production to CNF

- Unit production
  - A production $A \to B$ where the RHS consists of a single non-terminal symbol B
  - All other productions are non-unit productions (including $A \to a$ where a is a terminal symbol)

Algorithm to Convert a CFG with No ε-production to CNF

- Step 1
  - Construct a new set of productions P' by first including all non-unit productions
  - For non-terminal symbols A and B, if $A \to A1 \to A2 \to \cdots \to B$, (where $A1$, $A2$, … are non-terminal symbols), then for each non-unit production $B \to \alpha$, add a new production $A \to \alpha$ to P'

Algorithm to Convert a CFG with No ε-production to CNF

- Step 2
  - Consider each production in P' of the form $A \rightarrow X_1 X_2 \dots X_m$, where $m \geq 2$. If X_i is a terminal symbol a, introduce a new non-terminal symbol C and a production $C \rightarrow a$. Then replace X_i by C.

Algorithm to Convert a CFG with No ε-production to CNF

- Step 3
  - For each production in P' of the form $A \to B1\ B2\ B3$, replace this production by

    $A \to B1\ D1$

    $D1 \to B2\ B3$
  - For each production in P' of the form $A \to B1\ B2\ B3\ B4$, replace this production by

    $A \to B1\ E1$

    $E1 \to B2\ E2$

    $E2 \to B3\ B4$

  and so on. $D1$, $E1$, $E2$, etc. are new non-terminal symbols

Treebanks

- Treebank: A corpus in which each sentence is syntactically annotated with a parse tree
- Examples:
  - Penn Treebank (English, Arabic, Chinese)
  - Prague Dependency Treebank (Czech)

Penn Treebank

```
((S
   (NP-SBJ (DT That)
     (JJ cold) (, ,)
     (JJ empty) (NN sky) )
   (VP (VBD was)
     (ADJP-PRD (JJ full)
       (PP (IN of)
         (NP (NN fire)
           (CC and)
           (NN light) ))))
   (. .) ))
```

(a)

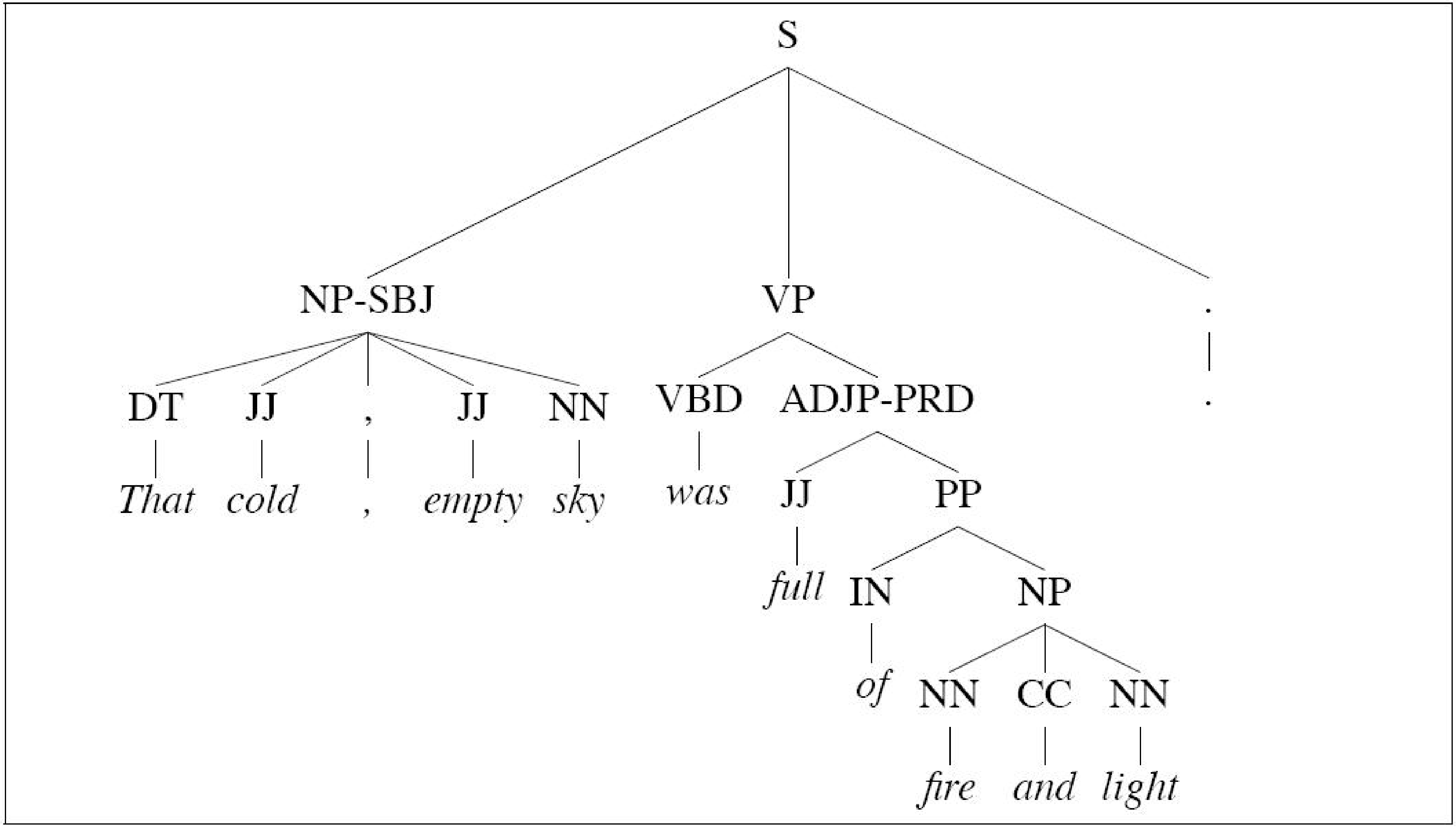
```
((S
   (NP-SBJ The/DT flight/NN )
   (VP should/MD
     (VP arrive/VB
       (PP-TMP at/IN
         (NP eleven/CD a.m/RB ))
       (NP-TMP tomorrow/NN )))))
```

(b)

Brown corpus

ATIS corpus

Penn Treebank



Constituency parse tree

Penn Treebank

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

" We would have to wait until we have collected on those assets , " he said .

Penn Treebank

- Traces (-NONE- nodes)
  - Mark long-distance dependencies or syntactic movement
  - Co-indexing
- Grammatical function tags
  - Denoted as “-X” after the main tag (where X is the grammatical function tag)
  - NP-SBJ (surface subject)
  - ADJP-PRD (non-VP predicate)

Heads and Head Finding

- Head: the word in the phrase that is grammatically the most important
- Heads are passed up a parse tree
- Each non-terminal in a parse tree is annotated with a word (its lexical head)
- Identify one RHS constituent of a CFG rule as the head child of that rule
- Lexical head of the parent (LHS of a CFG rule) is the lexical head of the head child

Heads and Head Finding

Head child (underlined):

S $\rightarrow$ NP <u>VP</u>

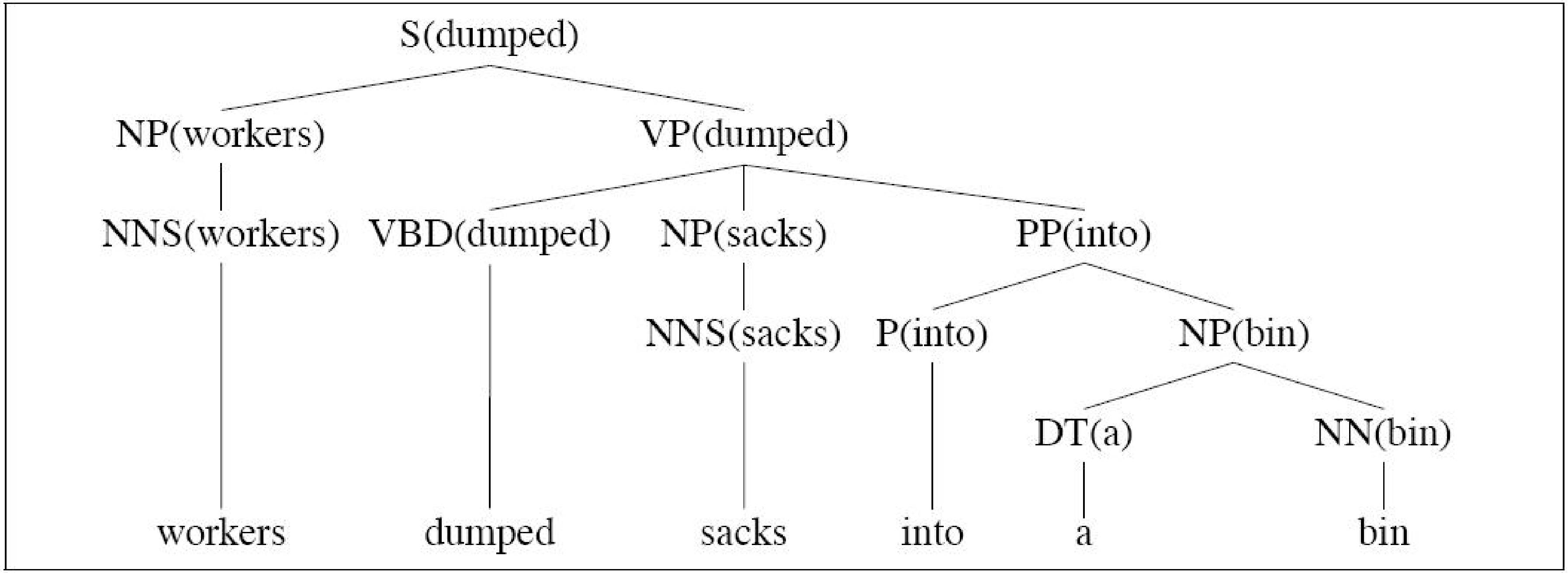
NP $\rightarrow$ <u>NNS</u>

VP $\rightarrow$ <u>VBD</u> NP PP

PP $\rightarrow$ <u>P</u> NP

NP $\rightarrow$ DT <u>NN</u>

Heads and Head Finding

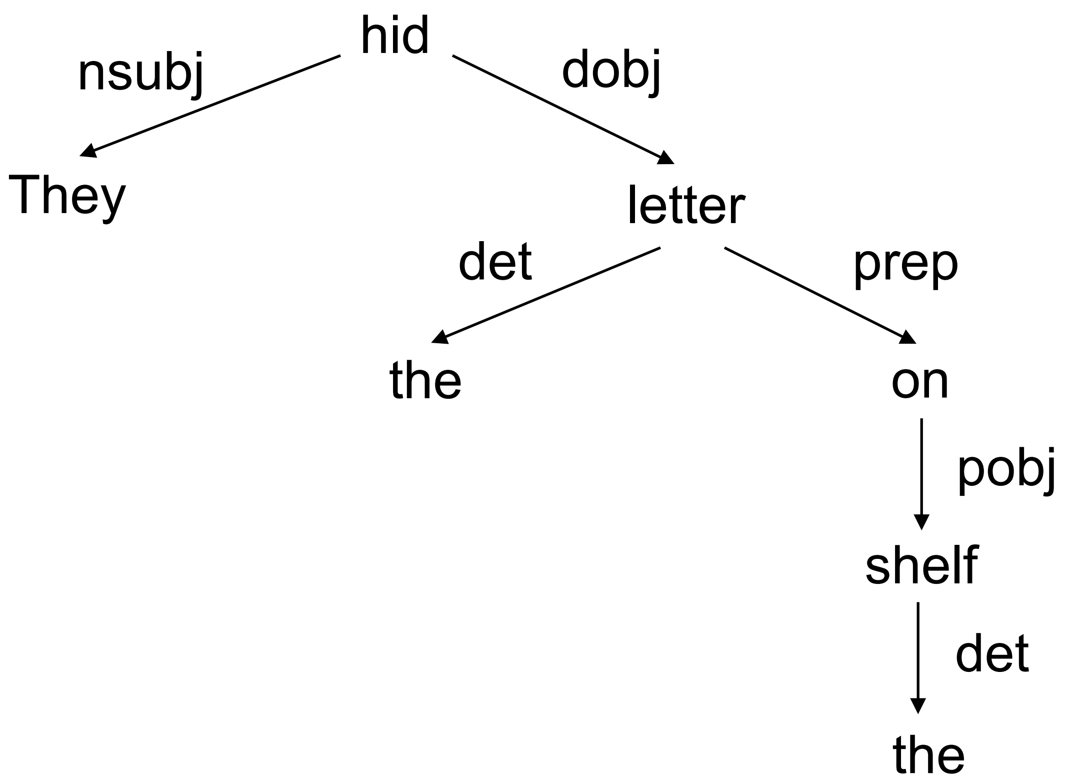


Dependency Grammars

- Binary relations between words in a sentence
- Ignore constituents and phrase structure rules (no non-terminal or phrasal nodes)
- Typed dependency parse
  - Links in parse tree are labeled (typed) from a fixed inventory of grammatical relations
- Untyped dependency parse
  - Links are unlabeled

Typed Dependency Parse

They hid the letter on the shelf



Grammatical Relations

Argument Dependencies	Description
nsubj	nominal subject
csubj	clausal subject
dobj	direct object
iobj	indirect object
pobj	object of preposition
Modifier Dependencies	**Description**
tmod	temporal modifier
appos	appositional modifier
det	determiner
prep	prepositional modifier

Dependency Grammars

- Advantages of dependency grammars
  - Strong predictive power that a word has on its dependents (e.g., a verb helps in deciding which noun is the subject or object)
  - Handle with ease languages with relatively free word order (e.g., Czech)

Converting Phrase Structure Parse to Untyped Dependency Parse

- Annotate the lexical head of each node in the phrase structure parse
- In the dependency parse, make the head of each non-head-child depend on the head of the head-child

Converting Phrase Structure Parse to Untyped Dependency Parse