# Natural Language Processing II (SC674)

Prof. Feng Zhiwei

# Ch. 2 Basic Concepts of Syntax

2 Basic Concepts of Syntax (6 hours)

2.1 Phrase Structure Grammar (PSG)

2.1.1 Algebraic definition of PSG

A PSG is a quadruple <V, Vt, S, P> such that

(1) V is a finite set of signs;
(2) Vt is a proper subset of V, called terminal symbols;
(3) S is a sign in V minus Vt, called start symbol, and
(4) P is a set of rewrite rules of the form $\alpha \rightarrow \beta$, where $\alpha$ is an element of $V^+$ and $\beta$ an element of $V^*$.

Here, $V^+$ is the positive closure (non-including empty element) and $V^*$ is the Kleene closure (including empty element) of V.

The basic components of PSG are the sets V, Vt and P, plus the start symbol S.

The terminal symbols of Vt are the word surfaces of the language.

The non-terminal symbols of V minus Vt are called the variables.

We will use Greek letters to represent the sequences from V*, upper case Latin letters to represent individual variables, and lower case Latin letters to represent individual terminal symbols.

A PSG generates language expressions by means of rewrite rule whereby the sign sequence on the left hand side (LHS) of a rule is replaced by the sign sequence on the right hand side (RHS). For example, if $\alpha \rightarrow \beta$ is a rewrite rule in P and $\gamma$, $\delta$ are sequences in $V^*$, then

$$\gamma \alpha \delta \rightarrow \gamma \beta \delta$$

is a direct substitution of the sequence $\gamma \alpha \delta$ by the sequence $\gamma \beta \delta$. In other words, by applying the rule $\gamma \alpha \delta \rightarrow \gamma \beta \delta$ to the sequence $\gamma \alpha \delta$ there results the new sequence $\gamma \beta \delta$.

The general format of rewrite rules in PSG suggests systematic restrictions of the following kind.

2.1.2 Restrictions of Phrase Structure rule (PS rule) schemata

-- Type 0: Unrestricted PS rules:

The left hand side and right hand site of a type 0 rule each consist of arbitrary sequences of terminal and non-terminal symbols.

■ Type 1: Context-sensitive PS rules

:

The left hand side and the right hand side of a type 1 rule each consist of arbitrary sequences of terminal and non-terminal symbols whereby the right hand side must be at least as long as the left hand side.

e.g. A B C $\rightarrow$ A D E C

- Type 2: Context-free PS rule

The left hand side of a type 2 rule consists of exactly one variable. The right hand side of the rule consists of a sequence from $V^+$.

e.g. A $\rightarrow$ BC
A $\rightarrow$ bBCc

- Type 3: Regular PS rules:

The left hand side of a type 3 rule consists of exactly one variable. The right hands side consists of exactly one terminal symbol and at most one variable..

e.g. A $\rightarrow$ b
A $\rightarrow$ bC

Because the rule types become more and more restrictive - from type 0 to type 3 – the rule of a certain type obey all restrictions of the lower rule types.
For example, the regular type 3 rule

A $\rightarrow$ bC

Complies with the lesser restrictions of the rule types 2, 1 and 0.
The context-free type 2 rule

A $\rightarrow$ BC

On the other hand, while not complying with the type 3 restriction, complies with the lesser restrictions of the lower rule types 1 and 0.

The different restriction of the PS-grammar rule schema results in four different types of PS grammar and PS language.

| Rule restrictions | types of PS-grammar | language classes |
|---|---|---|
| Type 3 | regular PSG | regular languages |
| Type 2 | context-free PSG | context-free language |
| Type 1 | context-sensitive PSG | context-sensitive languages |
| Type 0 | unrestricted PSG | recursive enumerative languages |

As an alternative to the PS-grammar hierarchy, also called Chomsky hierarchy.

### 2.1.3 Generative capacity and formal language classes

The PS-grammar type with the most restricted rules has the lower generative capacity.

#### 2.1.3.1 regular PS-grammar for $ab^k$ (k≥1)

V = {S, B, a, b}
Vt = {a, b}
P = { S $\rightarrow$ aB,

B $\rightarrow$ bB,

B $\rightarrow$ b}

The derivation tree of “abbbb”:

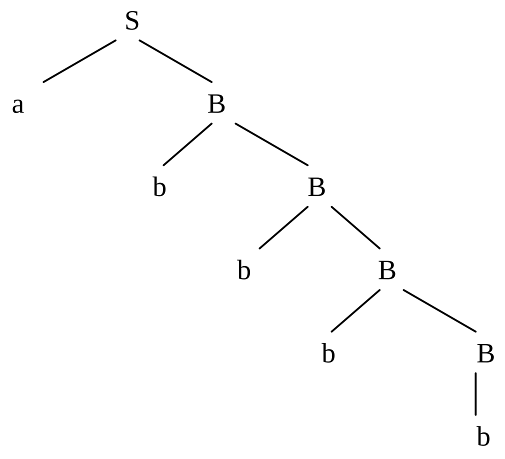


Fig. 1

2.1.3.2 regular PS-grammar for $\{a, b\}^+$

V = {S, a, b}

Vt = {a, b}

P = {S $\rightarrow$ aS,

S $\rightarrow$ bS,

S $\rightarrow$ a,

S $\rightarrow$ b}

The derivation tree of “abaaba”:

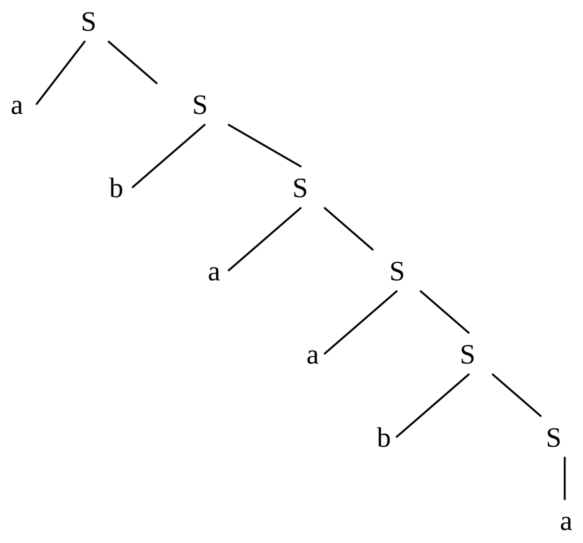


Fig. 2

2.1.3.3 regular PS-grammar for $a^m b^k$ (k, m $\geqslant$ 1)

V = {S, S1, S2, a, b}

Vt = {a, b}

P = {S $\rightarrow$ aS1,

S1 $\rightarrow$ aS1,

S1 $\rightarrow$ bS2

S2 $\rightarrow$ bS2,

S2 $\rightarrow$ b}

The derivation tree of "$a^3b^5$"

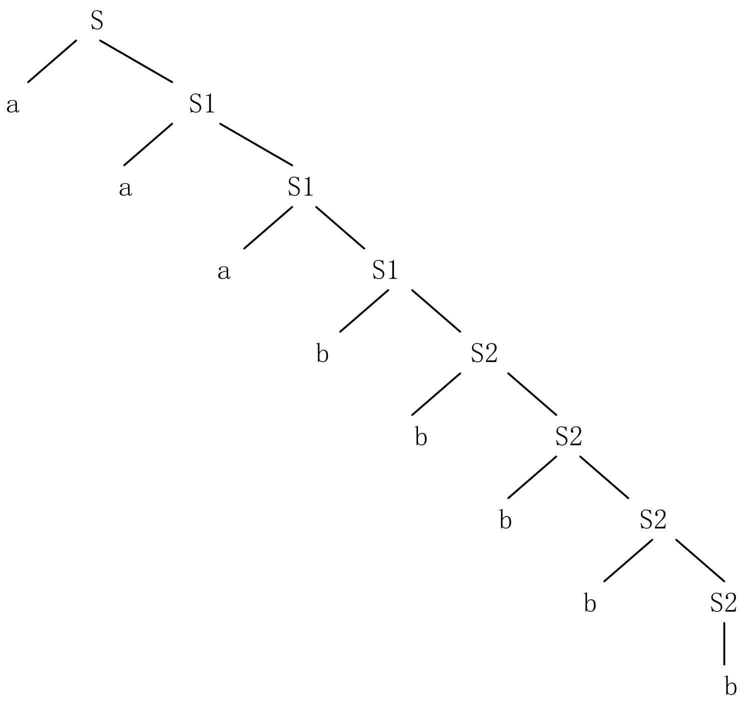


Fig. 3

The language $a^kb^k$ exceeds the generative capacity of a regular PS-grammar because it requires a correspondence between the number of the 'a' and the number of the 'b' (they must have the same superscript k).

2.1.3.4 context-free PS-grammar for $a^kb^k$ ($k \geqslant 1$)

V = {S, a, b}

Vt = {a, b}

P = {S $\rightarrow$ aSb,

S $\rightarrow$ ab}

The derivation tree of "$a^3b^3$" :

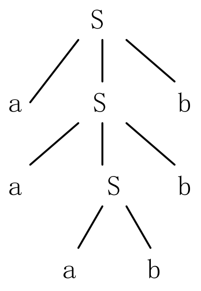


Fig. 4

This type of grammar is called as context-free grammar because the left hand side of type 2 rule consists by definition of a single variable, without a surrounding context of other signs.

2.1.3.5 context-free PS-grammar for $a^k b^{3k}$ (k ≥ 1)

V = {S, a, b}

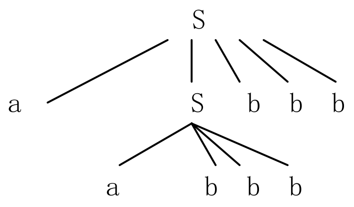
Vt = {a, b}

P = {S → aSbbb,

S → abbb}

The derivation tree of “aabbbbbb”

V = {S, a, b, c, d}:



2.1.3.6 context-free PS-grammar for $WW^R$

W represents an arbitrary sequences of words, e.g. abcd; and $W^R$ stands for the inverse sequence, e.g. dcba.

Context grammar is:

V = {S, a, b, c, d}

Vt = {a, b, c, d}

P = {S → aSa,

S → bSb,

S → cSc,

S → dSd,

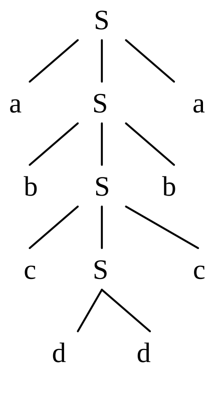
S → aa,

S → bb,

S → cc,

S → dd}

The derivation tree of abcddcba is:



2.1.3.7 PS-grammar for context-sensitive $a^k b^k c^k$:

V = {S, B, C, $D_1$, $D_2$, a, b, c}

Vt = {a, b, c}

P = {S → aSBC, 1

S → abC, 2

| | |
|---|---|
| $CB \rightarrow D_1B$, | 3a |
| $D_1B \rightarrow D_1D_{2,}$ | 3b |
| $D_1D_2 \rightarrow BD_{2,}$ | 3c |
| $BD_2 \rightarrow BC$, | 3d |
| $bB \rightarrow bb$, | 4 |
| $bC \rightarrow bc$, | 5 |
| $cC \rightarrow cc\}$ | 6 |

The rules $CB \rightarrow D_1B$

$D_1B \rightarrow D_1D_2$

$D_1D_2 \rightarrow D_1B$

$D_1B \rightarrow CB$

are all context-sensitive rules (e.g. "$CB \rightarrow D_1B$" means that in the context "~B", replace CB to $D_1B$) , they jointly have the same effect as the rule

$CB \rightarrow BC$

So this rule $CB \rightarrow BC$ is also a context-sensitive rule.
Rule 4, 5, 6 are all context-sensitive rules.
The derivation process of "aaabbbccc":

| Intermediate chains | rules |
|---|---|
| S | |
| $a\underline{S}BC$ | 1 |
| $a\underline{aSBC}BC$ | 1 |
| $aa\underline{abC}BCBC$ | 2 |
| $aaab\underline{BC}CBC$ | 3 |
| $aaabBC\underline{BC}C$ | 3 |
| $aaabB\underline{BC}CC$ | 3 |
| $aaa\underline{bb}BCCC$ | 4 |
| $aaab\underline{bb}CCC$ | 4 |
| $aaabb\underline{bc}CC$ | 5 |
| $aaabbb\underline{cc}C$ | 6 |
| $aaabbbc\underline{cc}$ | 6 |

The possibility of context-sensitively changing the order of sequences provides for a degree of control which is much higher than the context-free PS-grammar. The cost is the high degree of computational complexity. In order to correctly reconstruct automatically which sequence of context-sensitive rule applications resulted in a given expression, a potentially exponential number of reordering possibilities must be checked.
The largest language class in the PS-grammar hierarchy are the recursively enumerable languages, which are generated by the type 0 PS-grammars In type 0 PS-grammars, the right hand side of a rule may be shorter than the left hand side. This characteristic property of type 0 rule provides the possibility of deleting parts of sequences already generated.
For this reason, the class of recursively enumerable languages is un-decidable.

## 2.2 CFG for English

## 2.2.1 Context-Free Rules and Trees

Type 2 PS-grammar may be called as Context-Free Grammar (CFG). In the NLP, the Context-Free Grammar are also called Phrase-Structure Grammar (PSG). The formalism of CFG is equivalent to Backus-Naur Form (BNF).

The idea of basing a grammar on constituent structure dated back to the psychologist Wilhelm Wundt (1900), but was not formalized until Chomsky (1956), and, independently, Backus (1959).

We can use CFG to describe English. Following is an example. In formal grammar <V, Vt, S, P>, V is a finite set of signs; Vt is a proper subset of V, called terminal symbols. For natural language, the lexicon also is the sign, it will be too large if the lexicon also is included in V. In this case, we shall redefine the CFG grammar as follows:

$$G = <N, \Sigma, S, P>$$

Where N is a set of non-terminal symbols (or variables);

$\Sigma$ is a set of terminal symbols (disjoint from N);

S is designated start symbol;

P is a set of productions. Each of the form $A \rightarrow \alpha$, A is a non-terminal symbol and $\alpha$ is a string of symbols from the infinite set of strings $(\Sigma \cup N)^*$.

For the CFG rewriting rule "$A \rightarrow \alpha$", we say that A **directly derives** $\alpha$.

Let $\alpha_1 \rightarrow \alpha_2$, $\alpha_2 \rightarrow \alpha_3$, …, $\alpha_{m-1} \rightarrow \alpha_m$

We say that $\alpha_1$ **derives** $\alpha_m$, or $\alpha_1 => \alpha_m$.

We can define the language L(G) generated by a grammar G as the set of string composed of terminal symbols which can be derived from the start symbol S.

$L(G) = \{W \mid w \text{ is in } \Sigma^* \text{ and } S => w\}$

The problem of mapping from a string of words to its parse tree is called parsing.

For simplicity, we just write the part P in following CFG grammar.

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

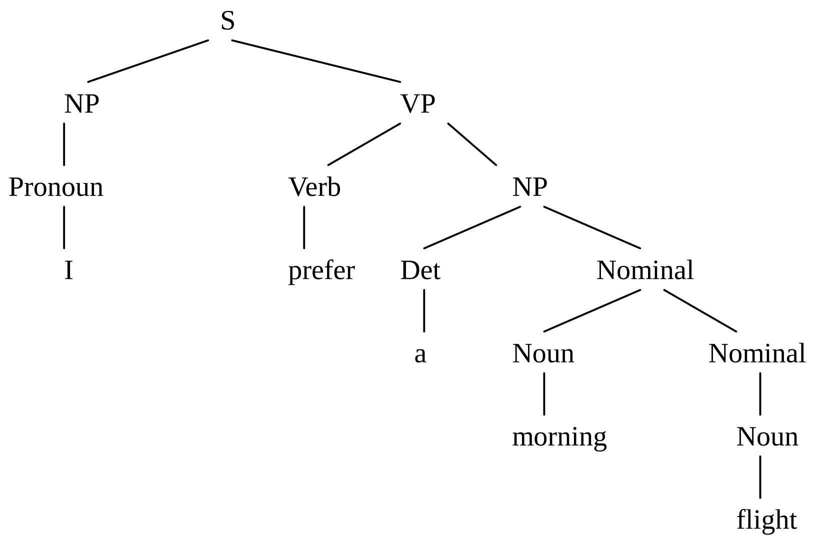
We can also write the rules for the lexicon.

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 | American | …
Determiner → the | a | an | this | these | that | ..
Preposition → from | to | on | near | …
Conjunction → and | or | but | …

The derivation tree of sentence "I prefer a morning flight" is as follows:



This derivation tree can be called as parse tree. We can also represent it using bracketed notation:

[S[NP[Pro I]][VP[V prefer][NP [Det a][Nom[N morning][Nom[N flight]]]]]]

2.2.2 Sentence Constructions of English

2.2.2.1 Declarative structure: A subject-NP followed by a VP (verb phrase)

S → NP VP

e.g. I prefer a morning flight
The flight should be eleven a.m. tomorrow
The return flight should leave at around seven p.m.
I want a flight from Ontario to Chicago

2.2.2.2 Imperative structure: It begins with a verb phrase, and have no subject.

S → VP

e.g. Show the lowest fare
Show me the cheapest fare that has lunch
List all flights between five and seven p.m.
Show me the last flight to leave

2.2.2.3 Yes-no-question structure: It begins with a auxiliary verb, followed by a subject-NP, followed by a VP.

S → Aux NP VP

e.g. Do any of these flights have stop?
Does LUFTHANSA flight eighteen twenty five serve dinner?
Can you give me the same information for France Airs?

2.2.2.4 Wh-question structure

2.2.2.4.1 Wh-subject-question structure

S → Wh-NP VP

e.g. What airlines fly from Seoul to Beijing?
Which flights serve breakfast?
Which of these flights have the longest stopover in Paris?

2.2.2.4.2 Wh-non-subject-question structure: The wh-phrase is not the subject of the sentence, and so the sentence includes another subject.

S → Wh-NP Aux NP VP

e.g. What flights do you have from Beijing to Seoul?

2.2.2.5 Other sentence structures:

There are other sentence structures in English we won't try to model here. For example, fronting, in which a phrase is placed at the beginning of the sentence for various discourse purposes (often involving topicalization and focus):

e.g. On Tuesday, I'd like to fly from Seoul to Tokyo.

2.2.3 The Noun Phrase

2.2.3.1 Before the Head Noun

NP → (Predet) (Det) (Card) (Ord) (Quant) (AP) Nominal
Predet: Predeterminer -- all
Det: Determiner – the a, an
Card: Cardinal number – one two
Ord: Ordinal number – first, second, other
Quant: Quantifier – many, several
() to mark optional constituents.
e.g. all the flights (Predet Det Nominal)
two friends (Card Nominal)
the first day (Det Ord Nominal)
the other flight (Det Ord Nominal)
many fares (Quant Nominal))
the longest layover (Det AP Nominal)

AP is the adjective phrase. AP can have an adverb before the adjective. So we can have the rules:

AP → (Adv) Adj
e.g. the least expensive fare

2.2.3.2 After the Head Noun

■ Nominal → Nominal PP (PP) (PP)

E.g a reservation [***on flight 937***] [***from Beijing***] [***to Frankfurt***]

-- Nominal → Nominal GerundVP
GerundVP → GerundV NP
| GerundV PP
| GerundV
| GerundV NP PP
GerundV → being | preferring | arriving | leaving | …
E,g any flight [arriving after nine a.m]

I need to have dinner ***reserved***

- Nominal → Nominal RelClause

  RelClause --. (who | that) VP

  e.g. flights ***that leave in the morning***

  Various postnominal modifiers can be combined.

  e.g. a flight [***form Beijing to Frankfurt***] [***leaving Monday morning***]

  a friend [***living in Seoul***] [***that would like to visit me here in Daejeon***]

2.2.4 Coordination

- NP → NP and NP

  e.g. Please repeat [NP[NP the flights] and [NP the costs]]
- VP → VP and VP

  e.g. What flights do you have [VP[VP leaving Seoul] and [VP arriving in San Francisco]]
- S → S and S

  e.g [S[S I'm interested in a flight from Beijing to Seoul] and [S I'am also interested in going to Busan]]

2.2.5 Agreement

In English, There is the agreement between the verb and the noun acting as its subject.

e.g. Do [NP any flights] stop in Seoul?

Does [NP this flight] stop on Seoul?

We can expand our grammar with multiple sets of rules, one rule set for 3sg (third person-singular) subjects, and one for non-3sg subjects.

For the rule

S → Aux NP VP

We could replace this with two rules of following form:

S → 3sgAux 3sgNP VP

S → Non-3sgAux Non3sgNP VP

Lexicon rules:

3sgAux → does | has | can | …

Non3sgAux → do | have | can | …

We also need to add rules for 3sgNP and Non3sgNP as follows (all lexical noun phrases can only be third person, so we use SgNominal to replace 3sgNominal, and use PlNominal to replace Non3sgNominal)::

3sgNP → (Det) (Card) (Ord) (Quant) (AP) SgNominal

Non3sgNP → (Det) (Card) (Ord) (Quant) (AP) PlNominal

SgNominal → SgNoun | SgNoun SgNoun

PlNominal → Plnoun | SgNoun PlNoun

SgNoun → flight | fare | dollar | reservation | …

PlNoun → flights | fares | dollars | reservations | …

We also need some rules to deal with first and second pronouns as follows.

1sgnomPronoun → I

1sggenPronoun → my

1sgaccPronoun → me

1plPronoun → we
2nomPronoun → you
2genPronoun → your
3sgnomPronoun → he | she | it
3sggenPronoun → his | her | its
3sgaccPronoun → him | her
3plnomnPronoun → they
3plgenPronoun → their

The difference on English pronoun in gender is not important in grammar because this difference can not lead to the inflection of verb.

A more significant problem occurs in languages like German, French, Russian, which not only have noun-verb agreement like English, but also have gender agreement. This adds another multiplier to the rule sets of the language. In this case, the description of CFG will become very complicated.

### 2.2.6 Verb Phrase and Sub-categorization

#### 2.2.6.1 Verb phrase

| | |
|---|---|
| VP → Verb | disappear, take off (the particle 'off' is considered to be an integral part of the verb) |
| VP → Verb NP | prefer a morning flight |
| VP → Verb NP PP | leave Seoul in the morning |
| VP → Verb PP | leaving on Monday |
| VP → Verb S | I [VP think [S I would like to take the ASIANA flight]] |
| VP → Verb that S | I [VPfind [that [S John slept]]] |

#### 2.2.6.2 Sub-categorization of verb

Here are some sub-categorization frame and example verbs

| Frame | Verb | Example |
|---|---|---|
| 0 | sleep | I want to sleep |
| NP | find | Find [NP the flight from Beijing to Seoul] |
| NP NP | show | Show [NP me] [NP airlines with flights from Seoul] |
| PPfrom PPto | fly | I would like to fly [PP from Beijing] [PP to Seoul] |
| NP PPwith | help | Can you help [NP me] [PP with a flight] |
| VPto | prefer | I would prefer [VPto to go by United airlines] |
| VPbrst | can | I can [VPbrst go from Seoul] |
| S | mean | Does this mean [S Beijing is the capital of China] |

VPbrst means bare stem verb VP

A single verb can take different sub-categorization frames. For example, "find" can take an NP NP frame (find me a flight) as well as an NP frame.

We may also represent the relation between verbs and their complements as follows:

Verb-with-NP-complement → find | leave | repeat | …
Verb-with-S-complement → think | believe | say | …
Verb-with-inf-VP-complement → want | try | need | …

Then each of our VP rules could be modified to require the appropriate verb subtype:

| | |
|---|---|
| VP → Verb-with-no-complement | disappear |
| VP → Verb-with-NP-comp NP | prefer a morning flight |
| VP → Verb-with-S-comp S | said there were two flights |

With this approach, the number of rules will increase vastly.

### 2.2.7 Auxiliaries

#### 2.2.7.1 Sub-categorization of auxiliaries:

- Modal: "can, could, may, might, must, will, would, shall, should". The head of VP is a bare stem. E.g. *can go in the morning*
- Perfect: "have". The head of VP is the past participle form. E.g. *have booked 3 flights*
- Progressive: "be". The head of VP is gerundive participle. E.g. *am going from Seoul.*
- Passive: The head of VP is the past participle. E.g. *was delayed by bad weather*

The head of VP including auxiliaries sub-categorize for particular kinds of VP complements. E.g. "can" would be listed in the lexicon as a verb-with-bare-stem-VP-complement.

#### 2.2.7.2 Multiple auxiliaries

The multiple auxiliaries in a sentence must occur in a particular order:

Modal < perfect < progressive < passive

| | |
|---|---|
| Modal perfect: | could have been a contender |
| Modal passive | will be married |
| Perfect progressive: | have been feasting |
| Modal perfect passive: | might have been prevented |

### 2.2.8 Grammar Equivalent and Normal Form

#### 2.2.8.1 Strong equivalence and weak equivalence

- Strong equivalence: Two grammars are strongly equivalent if they generate the same set of strings and if they assign the same phrase structure to each sentence (allowing merely for renaming of the non-terminal symbols).
- Weak equivalence: Two grammars are weakly equivalent if they generate the same set of strings but do not assign the same phrase structure to each sentence.

#### 2.2.8.2 Chomsky Normal form

A CFG is in Chomsky normal form (CNF) if it is $\varepsilon$ -free and if each production is either of the form A → B C or A → a. That is the RHS of each rule either has two non-terminal symbols or one terminal symbols. Chomsky normal form CFG grammars have binary trees.

Any grammar can be converted into weakly-equivalent Chomsky normal form grammar. E,g., a rule of the form

A → B C D

Can be converted into the following two CNF rules:

A → B X

X → C D

In CYK algorithm, every rule must be CNF.

### 2.2.9 Finite-State and Context-Free Grammar

#### 2.2.9.1 Recursion in NP

The recursion in a grammar occurs when expansion of a non-terminal includes the non-terminal itself.

E.g. Nominal → Nominal PP.

The finite-state grammar can not handles the recursion.

E.g. We have the CFG rule as follows:

NP → (Predet) (Det) (Card) (Ord) (Quant) (AP) Nominal

We could augment this rule by adding the PP as the post-modifier:

NP → (Predet) (Det) (Card) (Ord) (Quant) (AP) Nominal (PP)*

From PP → P NP, we could have:

NP → (Predet) (Det) (Card) (Ord) (Quant) (AP) Nominal (P NP)*

Now, NP is back in the NP rule!

From NP → (Predet) (Det) (Card) (Ord) (Quant) (AP) Nominal (P NP)*, we could have:

NP → (Predet) (Det) (Card) (Ord) (Quant) (AP) Nominal (P (Predet) (Det) (Card) (Ord) (Quant) (AP) Nominal P NP))*

Now, NP is back again in the NP rule !

We could also have the rule as follows:

NP → (Predet) (Det) (Card) (Ord) (Quant) (AP) Nominal (RelClause | GerundVP| PP)*

NP shall appears again in PP which was already included in NP rule !

If we would like use the finite state automate (FSA) to generate English sentence by these NP rules, the state number of FSA will become infinite.

Most of English NP rule is the recursion rule. The recursion cannot be handled in FSA.

#### 2.2.9.2 Center-embedded recursion

Chomsky (1959) proved that a context-free language L can be generated by a finite automaton if and only if there is a context-free grammar that generate L that does not have any center-embedded recursion (recursion of the form $A \rightarrow \alpha A \beta$ .

But above NP rules have the embedded recursion, so they cannot to be used to generate context-free language.

### 2.2.10 Grammar and Human Processing

Do people use CFG in their mental processing of language? It has proved very difficult to find clear-cut evidence that they do.

There are some interesting experiments:

- The experiment to examine the role of constituent in auditory comprehension by having subjects listen to sentences while also listening to short clicks at different times. Fodor and Bever (1965) found that subjects often mis-heard the clicks as if they occurred at constituent boundaries. They argued that the constituent was a “perceptual unit” which resisted interruption.
- Modularist and anti-modularist: There is a quite different disagreement about the human use of CFG. The modularists believe that human syntactic knowledge is a distinct module of the human mind, Anti-modularists believe that a complete model of syntactic structure will prove

to be impossible unless it includes knowledge from other domains (semantic, pragmatic, and social/interactional domains), the grammatical knowledge may incorporate semantic, pragmatic, and other constraints.

CFG rules for English

S → NP VP
S → VP
S → Aux NP VP
S → Wh-NP VP

NP → Pronoun
NP → Proper-Noun
NP → (predet) (Det) (Card) (Ord) (Quant) (AP) Nominal
NP → Nominal Noun

Nominal → Noun Nominal
Nominal → Noun
Nominal → Nominal PP (PP) (PP)
Nominal → Nominal GerundVP
    GerundVP → GerundV NP
    GerundVP → GerundV PP
    GerundVP → GerundV
    GerundVP → GerundV NP PP
Nominal → Nominal RelClause
    RelClause → who VP
    RelClause → that VP

AP → Adv Adj
AP → Adj

VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → Verb S
VP → Verb that S

PP → Prep NP

NP → NP and NP
VP → VP and VP
S → S and S