Chomsky Normal Form (CNF)

[Linz § 8.1, Kozen L. 22]

Definition: A Σ-CFG is in CNF if all its productions are of the form:

$$A \longrightarrow BC$$

or

$$A \longrightarrow a$$

where A, B, C are non-terminals and $a \in \Sigma$

Note:

No grammar in CNF can generate λ. (Why not?)

Further:

Theorem (Chomsky): For any CFG G, there is a CFG G^C in CNF s.t.

$$L(G^C) = L(G) \backslash \{\lambda\}$$

Proof : Given a CFG G: 2 steps

First: By $\boldsymbol{\lambda}$ and unit prod. elim. thm: (pp. 6-1/2)
Construct a CFG $\widehat{G}$ **without** $\boldsymbol{\lambda}$- or **unit** productions s.t.

$$L(\widehat{\boldsymbol{G}}) = L(G) \setminus \{\lambda\}.$$

Next: For each terminal symbol $\boldsymbol{a} \in \Sigma$,
add a **new non-terminal** A_a and a production:

$$A_a \to a.$$

Then **replace** each terminal $\boldsymbol{a}$ on the **rhs** of an old production (except a prod. of the form $B \to a$) by A_a.
Now all productions have the form:

$$A \to a$$

or

$$A \to B_1 B_2 \ldots B_k \quad (k \geq 2)$$

Finally: For each production

$$A \to B_1 B_2 ... B_k \quad (k \geq 3)$$

add a **new non-terminal** C, and **replace** the above production by:

$$A \to B_1 C$$

and

$$C \to B_2 B_3 ... B_k$$

Repeat until all **RHS's** of productions have length ≤ 2

The result is a grammar G^C in CNF s.t.

$$L(G^C) \; = \; L(\widehat{G}) \; = \; L(G) \backslash \{\lambda\}.$$

Back to our examples:

(1) $L(G) \; = \; \{a^n b^n \mid n \geq 0\}$.

By **λ/unit elim**, we get grammar $\widehat{G}$ (see p. 6-3)

with $$L(\widehat{\boldsymbol{G}}) \; = \; \{a^n b^n \mid n > 0\}$$

and productions: $S \to aSb, \quad S \to ab.$

Next, **add** non-terminals A, B, and **replace** above productions by:

$$S \to ASB, \quad S \to AB, \quad A \to a, \quad B \to b.$$

So we have prods

$$S \to ASB, \quad S \to AB, \quad A \to a, \quad B \to b.$$

Finally, **split up** the first production, i.e.

add non-terminal C, and **replace** $S \to ASB$ by:

$$S \to AC, \qquad C \to SB.$$

So the final grammar G^C in CNF is:

$$\underline{S \to AB}$$

$$\underline{S \to AC}$$

$$\underline{C \to SB}$$

$$\underline{A \to a}$$

$$\underline{B \to b}$$

(2) $L(G) = WN_{[\,]}$

G has productions:

$$S \to [\,S\,] \mid SS \mid \lambda.$$

By **λ/unit elim**, we get $\widehat{G}$ for $WN_{[\,]} \backslash \{\lambda\}$ with prods:

$$S \to [\,S\,] \mid SS \mid [\,]$$

Add non-terminals A, B, and **replace** above prods by:

$$S \to ASB \mid SS \mid AB$$

$$A \to [$$

$$B \to]$$

Finally, **add** new non-terminal C, and **replace** $S \to ASB$ by:

$$S \to AC$$

$$C \to SB$$

So the final grammar G^C for $WN_{[\,]}$ in CNF is:

$$\underline{S \to AB}$$

$$\underline{S \to AC}$$

$$\underline{S \to SS}$$

$$\underline{C \to SB}$$

$$\underline{A \to [}$$

$$\underline{B \to]}$$

Parse Trees for Chomsky Grammars

Given a CFL, L, with a **Chomsky grammar**, G, we want to investigate **parse trees** for words in L.

We will see: parse trees of Chomsky grammars are **"thin"**, i.e . **long** words derived in G must have **very long parse trees**.

This is because:

branching in such trees is (at most) **binary**.

For convenience, we deal with "**extended parse trees**," in which paths are continued downward past the leaves, repeating their terminal symbol.

Example: $L = \{a^n b^n \mid n \geq 0\}$

Chomsky grammar for L:

$$S \to AC \mid AB$$

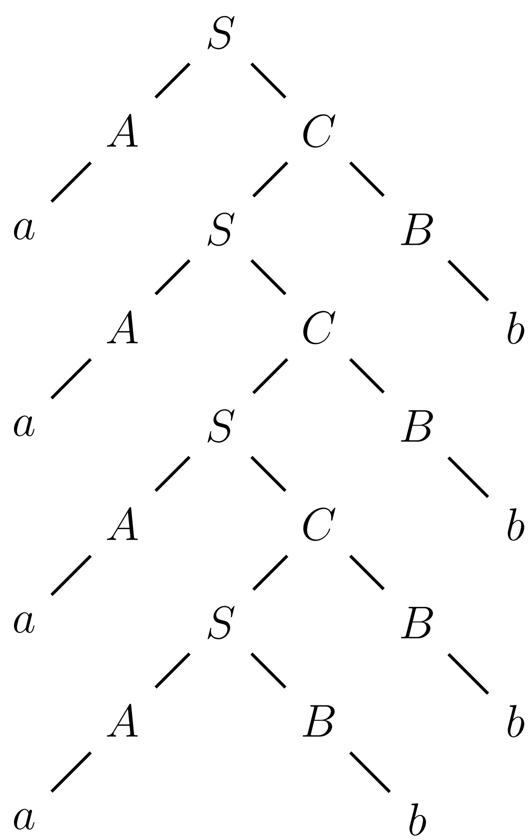
$$C \to SB$$

$$A \to a$$

$$B \to b$$

The parse tree for $w = a^4 b^4$:

[From Kozen, “Automata and Computability”, Lecture 22]



The **depth** of a word in a parse tree is the length of the **largest path** from the root to that word.

E.g. (on previous page) the depth of $\boldsymbol{a^4 b^4}$ is <u>8</u>

$Q.$ What is the depth of a parse tree for a word of length 2^n?

$A.$ Note that the width (i.e. # of nodes on one level) **at most doubles** at each level.

Therefore:

the width at level $0 = 1$, (i.e. the root)

the width at level $1 \leq 2$,

the width at level $2 \leq 4$,

$\vdots$

the width at level $n \leq \underline{2^n}$

<u>Conclusion</u>:

A word of **length** $\geq 2^n$, derived in a **Chomsky grammar**, needs a **parse tree** of **depth** $\geq n$.