

# Automata and Formal Languages

## CS138, Fall 2007

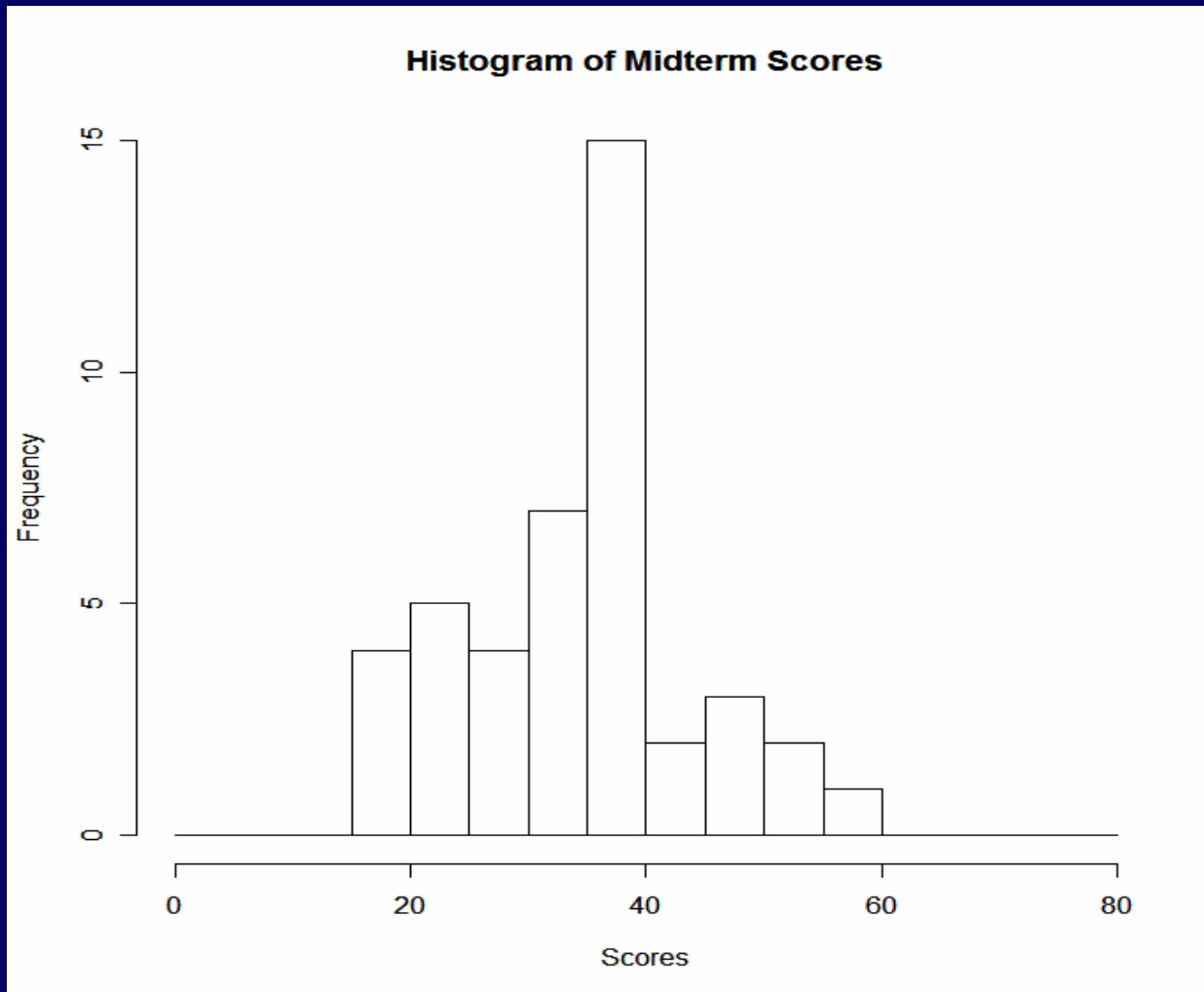
Wim van Dam

Room 2151, Harold Frank Hall

[vandam@cs.ucsb.edu](mailto:vandam@cs.ucsb.edu)

<http://www.cs.ucsb.edu/~vandam/>

# The Midterm Scores:



median: 36  
average: 35  
stdev: 10

Ballpark:  
A: 40-60  
B: 35-40  
C: 20-35

# Schedule

## This Week 8:

- Homework 6 will be announced this Wednesday.
- No class this Thursday, no Discussion this Friday.

## Next Week 9:

- Tuesday (W9) there will be a Discussion, no class.
- Homework 6 will be due next *Friday*.
- Homework 7 will be announced next week.
- Friday I will lecture on HW7 (instead of Discussion).

## Week 10:

- Loose ends before Final.

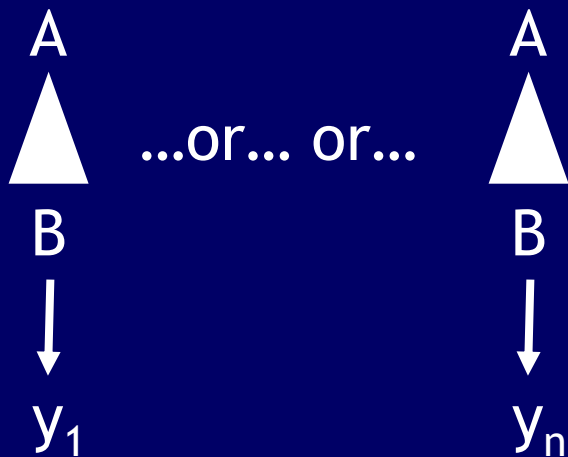
# Unit Productions

After removing useless rules and  $\lambda$ -productions, we also want to get rid of unit-productions of the kind  $A \rightarrow B$ .

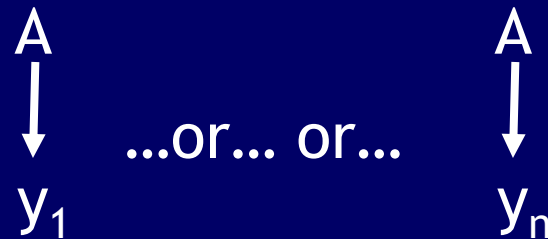
**Theorem 6.4:** For a  $\lambda$ -production free CFG, we can make an equivalent CFG without unit-productions. We do this again using the substitution rule.

# Trees of Unit Productions

If  $A \Rightarrow^* B$  is possible and we have the production rule  $B \rightarrow y_1 | \dots | y_n$ , we can have parts of the derivation trees that look like:



We want to remove the  $A \Rightarrow^* B$  part. Doing so requires to include new production rules of the kind  $A \rightarrow y_1 | \dots | y_n$ :



# Removing Unit Productions

Theorem 6.4: For a  $\lambda$ -production free CFG, we can make an equivalent CFG without unit-productions. We do this using the earlier described substitution rule.

Proof: Backtracking, collect all variable pairs  $(A,B)$  with  $A \Rightarrow^* B$  (ignoring the  $A \Rightarrow^* A$  cases).

First, add to  $P'$  all non-unit productions.

For all  $(A,B)$  with  $A \Rightarrow^* B$  and  $B \rightarrow y_1 | y_2 | \dots | y_n$  in  $P'$ , add to  $P'$  the productions:  $A \rightarrow y_1 | y_2 | \dots | y_n$ .

# Example 6.6

Take the CFG

$$S \rightarrow A0|B$$
$$B \rightarrow A|11$$
$$A \rightarrow 0|23|B$$

Unit productions for the CFG are:

$$S \Rightarrow^* B \quad \text{and} \quad S \Rightarrow^* A$$
$$B \Rightarrow^* A$$
$$A \Rightarrow^* B$$

For the CFG  $G'$ , we have  
the production rules:

$$S \rightarrow A0$$
$$B \rightarrow 11$$
$$A \rightarrow 0|23$$

...added with:

$$S \rightarrow 11 | 0 | 23$$
$$B \rightarrow 0 | 23$$
$$A \rightarrow 11$$

# Putting It All Together

Theorem 6.5: For all context free languages  $L$  without  $\lambda$ , there exist a context free grammar  $G$  that generates  $L$ , while  $G$  does not have useless productions,  $\lambda$ -productions, or unit productions.

Proof: *In the right order*, perform the manipulations:

1. Remove  $\lambda$ -productions (might produce unit-productions)
2. Remove unit-productions (does not create  $\lambda$ -productions)
3. Remove useless productions  
(does not create unit or  $\lambda$ -productions)

When parsing with such a grammar, every step ‘counts’.

# Normal Forms

We just saw how to transform a ( $\lambda$ -free) CFG into an equivalent CFG that has:

1. no  $\lambda$ -productions ( $A \Rightarrow^* \lambda$ )
2. no unit-productions ( $A \Rightarrow^* B$ )
3. no useless variables or useless productions

Next we will discuss two important normal forms: the Chomsky Normal Form and the Greibach Normal Form, and the fast parsing of CFGs in CNF [Reader, pp. 80-84].

# Chomsky Normal Form

Definition 6.4: A CFG is in Chomsky normal form if and only if all production rules are of the form

$$A \rightarrow BC$$

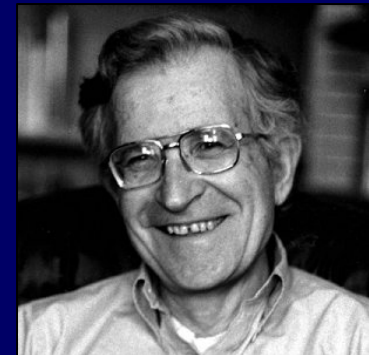
or  $A \rightarrow x$

with variables  $A, B, C \in V$  and  $x \in T$ .

(Sometimes rule  $S \rightarrow \lambda$  is also allowed.)

CFGs in CNF can be parsed in time  $O(|w|^3)$ .

Named after Noam Chomsky who in the 60s made seminal contributions to the field of theoretical linguistics. (cf. Chomsky hierarchy of languages).



# Theorem 6.6

Theorem 6.6: Every  $\lambda$ -free CFG  $G$  can be described by an equivalent CFG  $G'$  in Chomsky normal form. The transformation from  $G$  to  $G'$  can be done efficiently.

## Outline of Proof:

1. Eliminate all unit,  $\lambda$  and useless productions of  $G$ .
2. Rewrite such that all terminal producing rules are of the form  $B_a \rightarrow a$ .
3. Rewrite such that all variable producing rules are of the form  $A \rightarrow CD$  with  $C, D \in V$ .

# Details of Proof

Step 2: How do you transform general production rules of the kind  $A \rightarrow y_1 \dots y_n$  with  $y_j \in V \cup T$  to rules that are of the kind  $A \rightarrow y_1 \dots y_n$  with  $y_j \in V$  or  $A \rightarrow y$  with  $y \in T$ ?

Answer: Introduce dedicated terminal producing variables  $B_y \rightarrow y$  for each  $y \in T$  and replace  $y$  by  $B_y$ .

Step 3: How do you transform production rules of the kind  $A \rightarrow C_1 \dots C_n$  with  $C_j \in V$  to rules of the kind  $A \rightarrow C_1 C_2$ ?

Answer: Make a chain of rules to produce  $C_1 \dots C_n$ :  
 $A \rightarrow C_1 D_1$  and  $D_1 \rightarrow C_2 D_2$  and ... and  $D_{n-2} \rightarrow C_{n-1} C_n$ .

# Example of Turning into CNF

- Initial grammar:  $S \rightarrow 0S1 \mid AAA$  and  $A \rightarrow 0 \mid SA$

Create a,b terminal producing variables X and Y to get:

$$S \rightarrow XSY \mid AAA$$

$$A \rightarrow 0 \mid SA$$

$$X \rightarrow 0$$

$$Y \rightarrow 1$$

Make variable chains to get:

$$S \rightarrow XS_1 \mid AS_2$$

$$S_1 \rightarrow SY$$

$$S_2 \rightarrow AA$$

$$A \rightarrow 0 \mid SA$$

$$X \rightarrow 0$$

$$Y \rightarrow 1$$

Note that we did *not* create the unit rule  $A \rightarrow X$ .

# CFG Membership Algorithm

The CYK algorithm (Cocke-Younger-Kasami) decides in time  $O(|w|^3)$  whether or not  $w \in L(G)$  with  $G$  in Chomsky NF.

How it works: Let the string be  $w = a_1 \dots a_n$  and define  $V_{ik} = \{ A \in V : A \Rightarrow^* a_i \dots a_k \}$  for all  $1 \leq i \leq k \leq n$ , so that we want to know “ $S \in V_{1n}$ ?”

We solve this by first determining  $V_{11}, V_{22}, \dots, V_{nn}$ , then  $V_{12}, V_{23}, \dots, V_{n-1 n}$ , then  $V_{13}, \dots$ , up to the final  $V_{1n}$ .

Observation 1: Because of CNF, finding the  $V_{ij}$  is trivial.

Observation 2: Also,  $V_{ik}$  is determined by the combinations

$$V_{ik} = \{ A \in V : A \rightarrow BC \text{ with } B \in V_{ij} \text{ and } C \in V_{j+1 k} \text{ and } i \leq j \leq k \}.$$

Using this *dynamic programming technique* we find  $V_{1n}$ .

# CYK in Action

Take the CNF grammar

$$S \rightarrow AB \mid CC$$

$$A \rightarrow CC$$

$$B \rightarrow BC \mid 0$$

$$C \rightarrow 0 \mid 1$$

with  $w = 1101 \in L$ ?

- $V_{11} = \{C\}, V_{22} = \{C\}, V_{33} = \{B, C\}, V_{44} = \{C\}$

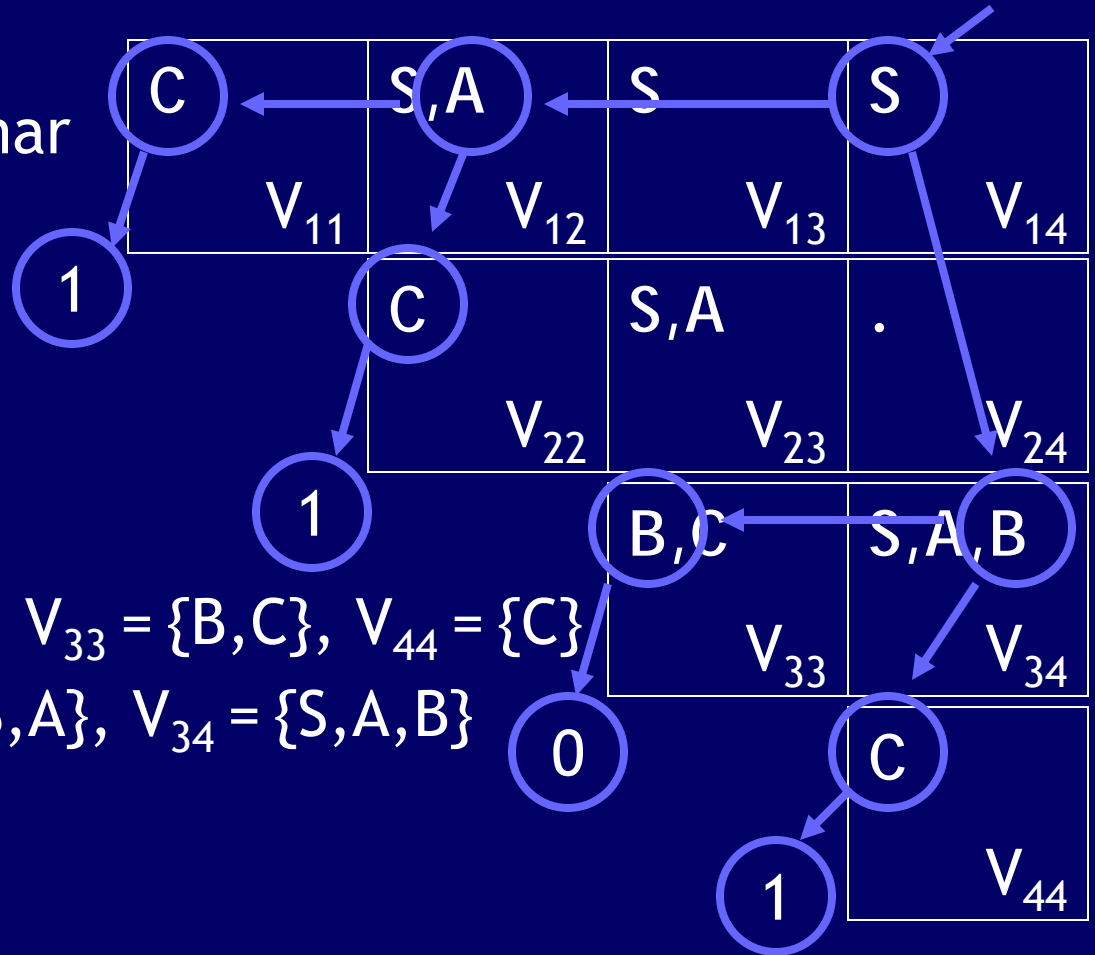
- $V_{12} = \{S, A\}, V_{23} = \{S, A\}, V_{34} = \{S, A, B\}$

- $V_{13} = \{S\}, V_{24} = \{\}$

- $V_{14} = \{S\}$

Retracing the  $V_{14} = \{S\}$  result gives the derivation tree:

- $S \Rightarrow AB \Rightarrow CCB \Rightarrow 1CB \Rightarrow 11B \Rightarrow 11BC \Rightarrow 110C \Rightarrow 1101$



# Complexity of CYK

There are  $O(n^2)$  variable sets  $V_{ik}$  that we have to construct.

For each set  $V_{ik}$  there are no more than  $n$  pairs  $(V_{ij}, V_{j+1 k})$  that we have to consider to determine  $V_{ik}$ .

In total, the running time is upper bounded by  $O(n^3)$ .

Note that this does *not* include the time required to bring the CFG into Chomsky Normal Form (which also can be done efficiently though).