

# CS\_275 Automata and Formal Language Theory

Course Notes

Additional Material

(This material is no longer taught and not exam relevant)

Part II: The Recognition Problem (II)

Sect II.5.: Context Free Grammars and Programming Languages (14)

Anton Setzer

(Based on a book draft by J. V. Tucker and K. Stephenson)

Dept. of Computer Science, Swansea University

[http://www.cs.swan.ac.uk/~csetzer/lectures/  
automataFormalLanguage/current/index.html](http://www.cs.swan.ac.uk/~csetzer/lectures/automataFormalLanguage/current/index.html)

II.5.1. Derivation Trees for Context-Free Grammars (14.1)

II.5.2. Uniqueness of Derivation Trees (14.1)

II.5.3. Normal Forms for Context-Free Grammars (14.2)

II.5.4. The Pumping Lemma for CFG (14.4)

II.5.5. Floyd's Theorem

## II.5.1. Derivation Trees for Context-Free Grammars (14.1)

### II.5.2. Uniqueness of Derivation Trees (14.1)

### II.5.3. Normal Forms for Context-Free Grammars (14.2)

### II.5.4. The Pumping Lemma for CFG (14.4)

### II.5.5. Floyd's Theorem

# Notations for Derivation Trees

We introduce the following notations:

- ▶ If  $D$  is a derivation tree, then
  - ▶ label( $D$ ) is the label of the root of  $D$ .
  - ▶ children( $D$ ) =  $[D_1, \dots, D_n]$  means that the children (immediate subtrees of  $D$ ) are  $D_1, \dots, D_n$  (read from left to right).
  - ▶ frontier( $D$ ) denotes the frontier of  $D$ .
- ▶ If  $D_1, \dots, D_n$  are derivation trees,  $A$  a nonterminal, let  $D :=$  tree( $A, D_1, \dots, D_n$ ) be the derivation tree s.t.
  - ▶ label( $D$ ) =  $A$ ,
  - ▶ children( $D$ ) =  $[D_1, \dots, D_n]$ .
- ▶ A special derivation tree is the tree with only one node, namely the root and no subtrees for this node. It is called the trivial derivation tree.

# Inductive Definition of Derivations

We can define the set of derivation trees as well inductively as follows:

## Definition

We define the set of derivation trees  $D$  for a CFG  $G = (T, N, S, P)$  inductively together with

- ▶ label( $D$ )  $\in T \cup N \cup \{\epsilon\}$ ,
- ▶ children( $D$ ), a list of derivation trees,
- ▶ frontier( $D$ )  $\in (T \cup N)^*$ .

as follows:

- ▶ If  $A \in T \cup N \cup \{\epsilon\}$ , then
  - ▶  $D := \text{tree}(A)$  is a derivation tree, (the trivial derivation tree),
  - ▶  $\text{label}(D) := A$ ,
  - ▶  $\text{children}(D) = []$ ,
  - ▶  $\text{frontier}(D) := A$ .

# Inductive Definition of Derivations

We can define the set of derivation trees as well inductively as follows:

## Definition (Cont)

- ▶ If  $A \in N$ ,  $D_1, \dots, D_n$  are derivation trees,

$A \rightarrow \text{label}(D_1).\text{label}(D_2).\dots.\text{label}(D_n)$  is a production

and  $n = 1 \vee \forall i.\text{label}(D_i) \neq \epsilon$ , then

- ▶  $D := \text{tree}(A, D_1, \dots, D_n)$  is a derivation tree,
- ▶  $\text{label}(D) := A$ ,
- ▶  $\text{children}(D) := [D_1, \dots, D_n]$ ,
- ▶  $\text{frontier}(D) := \text{frontier}(D_1).\text{frontier}(D_2).\dots.\text{frontier}(D_n)$ .

## Size of a Derivation Tree

We want show that for every derivation tree we can find a corresponding derivation and vice versa. For this we need a measure of the size of derivation tree.

### Definition

The size  $\text{size}(D)$  of a derivation tree  $D$  is defined by induction on the definition of trees:

- ▶ If  $D = \text{tree}(x)$ , then  $\text{size}(D) := 1$ .
- ▶ If  $D = \text{tree}(A, D_1, \dots, D_n)$ , then  $\text{size}(D) := 1 + \text{size}(D_1) + \dots + \text{size}(D_n)$ .

We define as well the height  $\text{height}(D)$  of a derivation tree:

- ▶ If  $D = \text{tree}(x)$ , then  $\text{height}(D) := 1$ .
- ▶ If  $D = \text{tree}(A, D_1, \dots, D_n)$ , then  $\text{height}(D) := 1 + \max\{\text{size}(D_1), \dots, \text{size}(D_n)\}$ .

# Proof of Theorem II.5.1.1.

Theorem II.5.1.1. follows by Lemma II.5.1.2. below.

We first introduce the notion of a derivation forest, which generalises derivation trees.



# Derivation Forest

For proving the equivalence of derivation trees and derivations, we need to deal with derivations  $w \Rightarrow^* w'$  where  $w$  is a string. Such derivations correspond to derivation forests, as defined as follows:

## Definition

Let  $G = (T, N, S, P)$  be a CFG,  $w, w' \in (T \cup N)^*$ ,  $w \neq \epsilon$ . Let  $w = x_1, \dots, x_l$ ,  $x_i \in T \cup N$ .

A derivation forest with root  $w$  and frontier  $w'$  is a list of derivations  $[D_1, \dots, D_l]$  s.t.

- ▶  $\text{label}(D_i) = x_i$ ,
- ▶  $\text{frontier}(D_1).\text{frontier}(D_2).\dots.\text{frontier}(D_l) = w'$ .

Furthermore  $\text{size}([D_1, \dots, D_l]) := \text{size}(D_1) + \dots + \text{size}(D_l)$ .

## Lemma II.5.1.2. (Derivation Trees and Language Generation)

### Lemma (II.5.1.2.)

Let  $G = (T, N, S, P)$  be a CFG,  $A \in T$ ,  $w, w' \in (T \cup N)^*$ , Then the following are equivalent

- (1) There exist a derivation forest  $D$  with root  $w$  and frontier  $w'$ .
- (2)  $w \Rightarrow^* w'$ .

In case  $w' \in T^*$ , the derivation sequence  $w \Rightarrow^* w'$  can both be chosen as a left-most and as a right-most derivation sequence

Proof (1)  $\Rightarrow$  (2)

- ▶ The proof is by induction on  $\text{size}(D)$ .
- ▶ We will do it in such a way that we get, in case all leaves are terminal symbols, a left-most derivation.
- ▶ A right most derivation can be obtained by choosing instead of the left-most the right most non-trivial derivation tree.
- ▶ Let  $w = x_1 \cdots x_n$ ,  $D = [D_1, \dots, D_n]$ .
- ▶ If all  $D_i$  are trivial, then  $w' = w$ ,  $w \Rightarrow^* w'$ .
- ▶ So assume at least one  $D_i$  is non-trivial. Let  $D_k$  be the left-most non-trivial derivation tree, i.e.  $D_1, \dots, D_{k-1}$  are trivial,  $D_k$  is non-trivial.
- ▶ Let  $D_k = \text{tree}(A, D'_1, \dots, D'_l)$ ,  $\text{label}(D'_i) = x'_i$ .  $x_k = A$ .

Proof (1)  $\Rightarrow$  (2)

- ▶ Then

$$A \longrightarrow x'_1 \cdots x'_l$$

is a production, and we have that with

$$w_1 := x_1 \cdots x_{k-1} x'_1 x'_2 \cdots x'_l x_{k+1} x_{k+2} \cdots x_n$$

that

$$w = x_1 \cdots x_{k-1} A x_{k+1} \cdots x_n \Rightarrow w_1$$

is a one-step derivation.

In case  $w' \in T^*$ , we have  $x_i \in T$ , and this one-step derivation was left-most.

- ▶ We have that

$$[D_1, \dots, D_{k-1}, D'_1, \dots, D'_l, D_{k+1}, D_{k+2}, \dots, D_n]$$

is a derivation forest with root  $w_1$  and frontier  $w'$ , and has size  $\text{size}(D) - 1$ .

Proof (1)  $\Rightarrow$  (2)

- ▶ By IH there exist a derivation  $w_1 \Rightarrow^* w'$ , which in case of  $w' \in T^*$  can be chosen as a left-most derivation sequence.
- ▶ Therefore  $w \Rightarrow w_1 \Rightarrow^* w'$  is a derivation, which in case of  $w' \in T^*$  can be chosen as a left-most derivation sequence.

(2)  $\Rightarrow$  (1)

- ▶ Proof is by induction on the length of the derivation  $w \Rightarrow^* w'$ .
- ▶ Let  $w = x_1, \dots, x_n$ .
- ▶ In case the length is 0,  $w' = w$ , and we can choose  $D = [\text{tree}(x_1), \dots, \text{tree}(x_n)]$ .
- ▶ Otherwise, assume that  $x_k = A$  is a non-terminal,  $A \rightarrow y_1 \cdots y_l$  (with  $y_i \in T \cup N$  or  $l = 1 \wedge y_1 = \epsilon$ ). Let

$$w_1 := x_1 \cdots x_{k-1} A x_{k+1} \cdots x_n \Rightarrow x_1 \cdots x_{k-1} y_1 y_2 \cdots y_l x_{k+1} \cdots x_n$$

- ▶ Assume that the derivation is

$$w = x_1 \cdots x_{k-1} A x_{k+1} \cdots x_n \Rightarrow w_1 \Rightarrow^* w'$$

- ▶ By IH there exist a derivation forest

$$[D_1, \dots, D_{k-1}, D'_1, \dots, D'_l, D_{k+1}, \dots, D_n]$$

with root  $w_1$  and frontier  $w'$ .

$(2) \Rightarrow (1)$ 

► Now

$$[D_1, \dots, D_{k-1}, (A, D'_1, \dots, D'_l), D_{k+1}, \dots, D_n]$$

is a forest with root  $w$  and frontier  $w'$ .

II.5.1. Derivation Trees for Context-Free Grammars (14.1)

II.5.2. Uniqueness of Derivation Trees (14.1)

II.5.3. Normal Forms for Context-Free Grammars (14.2)

II.5.4. The Pumping Lemma for CFG (14.4)

II.5.5. Floyd's Theorem



## Lemma II.5.2.3. Uniqueness of Derivation (Trees)

### Lemma (II.5.2.3.)

Let  $G = (T, N, S, P)$  be a CFG,  $A \in N$ ,  $w \in T^*$ .

- (1) *Assume there are two different derivation trees with root labelled by  $A$  and frontier  $w$ . Then there exist two different left-most and two different right-most derivations of  $A \Rightarrow^* w$ .*
- (2) *Assume there are two different left-most derivations or two different right-most-derivations of  $A \Rightarrow^* w'$ . Then there exist two different derivation trees of with root labelled by  $A$  and frontier  $w$ .*

## Proof of Lemma II.5.2.3

- ▶ The example given in the general material demonstrates how derivation trees are transformed into left-most derivations in a unique way.
- ▶ A more formal proof is carried out by proving the following more general Lemma II.5.2.4:

## Lemma II.5.2.4. Uniqueness of Derivation (Trees)

### Lemma (II.5.2.4.)

Let  $G = (T, N, S, P)$  be a CFG,  $w \in (T \cup N)^+$ ,  $w' \in T^*$ .

- (1) Assume there are two different derivation forests with root  $w$  and frontier  $w'$ . Then there exist two different left-most and two different right-most derivations of  $w \Rightarrow^* w'$ .
- (2) Assume there are two different left-most derivations or two different right-most-derivations of  $w \Rightarrow^* w'$ . Then there exist two different derivation forests of with root  $w$  and frontier  $w'$ .

# Proof of the Lemma II.5.2.4. (1)

- ▶ We prove only that there exist two different left-most derivations.
- ▶ Let  $[D_1, \dots, D_n]$  and  $[D'_1, \dots, D'_n]$  be two different derivation forests with root  $w$  and frontier  $w'$ .
- ▶ Induction on the length of the first derivation forest.
- ▶ If all  $D_i$  are trivial, then  $w \in T^*$ ,  $w = w'$ , but then  $D'_i = D_i$ , which is not possible.
- ▶ Let  $D_k$  be the first non-trivial derivation tree, i.e.  $D_1, \dots, D_{k-1}$  are trivial.
- ▶ Let  $D_k = \text{tree}(A, D''_1, \dots, D''_l)$ ,  $\text{label}(D''_i) = y_i$ ,  $A \rightarrow y_1 \cdots y_l$  a production.
- ▶ Let  $w_1 := x_1 \cdots x_{k-1} y_1 \cdots y_l x_{k+1} \cdots x_n$ .
- ▶ We have  $D'_i$  are trivial for  $i < k$ ,  $D_k$  must be non-trivial (since it has as label a nonterminal).

## Proof of the Lemma II.5.2.4. (1)

- ▶ Case 1:  $D'_k$  has the same production at the root, i.e.  
 $D'_k = \text{tree}(A, D'''_1, \dots, D'''_l)$ ,  $\text{label}(D'''_i) = y_i$ .

- ▶ Then

$$[D_1, \dots, D_{k-1}, D''_1, \dots, D''_l, D_{k+1}, \dots, D_n]$$

and

$$[D'_1, \dots, D'_{k-1}, D'''_1, \dots, D'''_l, D'_{k+1}, \dots, D'_n]$$

must be different.

- ▶ By IH there exist two different left-most productions  $w_1 \Rightarrow w'$ .
- ▶ Therefore we obtain two different left-most productions  $w \Rightarrow w_1 \Rightarrow w'$ .

# Proof of the Lemma II.5.2.4. (1)

- ▶ Case 2:  $D'_k$  has a different production at the root, i.e.  
 $D'_k = \text{tree}(A, D'''_1, \dots, D'''_m)$ ,  $\text{label}(D'_i) = y'_i$ .  $y'_1 \cdots y'_m \neq y_1 \cdots y_m$ .
  - ▶ But then the first steps in the derivations constructed in the lemma are different, and we obtain two different left-most derivations.

## Proof of the Lemma II.5.2.4. (2)

This proof is similar, at the first place where the two derivations differ we construct two different derivation forests.

## Theorem II.5.2.4 Uniqueness of Derivation (Trees)

A direct consequence of the theorem is the following:

### Theorem (II.5.2.4)

Let  $G = (T, N, S, P)$  be a CFG,  $w \in T^*$ . The following are equivalent:

- (1) There exist exactly one derivation tree with label  $S$  and frontier  $w$ .
- (2) There exist exactly one left-most derivation sequence  $S \Rightarrow^* w$ .
- (3) There exist exactly one right-most derivation sequence  $S \Rightarrow^* w$ .



II.5.1. Derivation Trees for Context-Free Grammars (14.1)

II.5.2. Uniqueness of Derivation Trees (14.1)

II.5.3. Normal Forms for Context-Free Grammars (14.2)

II.5.4. The Pumping Lemma for CFG (14.4)

II.5.5. Floyd's Theorem

# Chomsky Normal Form

## Definition

A CFG  $G = (T, N, S, P)$  is in Chomsky Normal Form if all of its productions are

either of the form  $A \rightarrow BC$  or of the form  $A \rightarrow a$

where  $A, B, C \in N$  and  $a \in T$ .

# Chomsky Normal Form

## Remark

*If  $G$  is a CFG in Chomsky Normal Form, then  $\epsilon \notin L(G)$ .*

Proof: Let  $G = (T, N, S, P)$ .

Since  $G$  has no production  $A \rightarrow \epsilon$ , we have that if  $w \Rightarrow w'$  then the  $|w'| \geq |w|$ .

Therefore if  $w \Rightarrow^* w'$  then  $|w'| \geq |w|$ .

Therefore, if  $w \in L(G)$  then  $S \Rightarrow^* w$  therefore  $|w| \geq |S| = 1$ ,  $w \neq \epsilon$ .

# Chomsky Normal Form

We are going to prove the following:

## Theorem

*For any CFG  $G$  there exist a CFG  $G'$  in Chomsky Normal Form s.t.*

$$L(G') = L(G) \setminus \{\epsilon\}$$

# Step 1: Removal of null productions

## Definition

A null-production of a grammar  $G = (T, N, S, P)$  is a production of the form  $A \rightarrow \epsilon$ .

## Lemma

*If  $G$  is a CFG. Then there exist a CFG  $G'$  which doesn't have any null productions, and s.t.  $L(G') = L(G) \setminus \{\epsilon\}$ .*

# Proof

Let  $G = (T, N, S, P)$ .

We first define the set of nullable nonterminals of  $G$ .

A nonterminal  $A$  is nullable if  $A \rightarrow \epsilon$ .

They can be defined as follows:

- ▶ If  $A \rightarrow \epsilon$ , then  $A$  is nullable.
- ▶ If  $A \rightarrow B_1 \cdots B_k$  for nullable nonterminals  $B_1, \dots, B_k$ , then  $A$  is nullable.

# Obtaining $G'$ from $G$

We obtain the grammar  $G' = (T, N, S, P')$  from  $G = (T, N, S, P)$ , by defining  $P'$  from  $P$  as follows:

- ▶ We start by taking all productions of  $P$ .
- ▶ We remove all productions  $A \rightarrow \epsilon$  from  $P$ .
- ▶ If  $A \rightarrow w$  is a production in  $P$ , and  $w'$  is obtained by omitting some but not all occurrences of nullable nonterminals from  $w$ , then

$$A \rightarrow w'$$

is added to  $P$ .

# Equivalence of $G$ and $G'$

- ▶ We show that  $L(G') = L(G) \setminus \{\epsilon\}$ .
- ▶ First  $L(G') \subseteq L(G)$ , because if  $A \rightarrow w'$  is a new production added to  $P'$ , then  $A \Rightarrow_G^* w'$ , and therefore from any derivation in  $G'$  we obtain a derivation in  $G$ .



# Equivalence of $G$ and $G'$

- ▶  $L(G) \setminus \{\epsilon\} \subseteq L(G')$ :
  - ▶ We show that from a derivation tree  $D$  of  $G$  with  $\text{frontier}(D) \neq \epsilon$  we obtain a derivation tree  $D'$  of  $G'$  with the same frontier and the same root by induction on the derivations.
  - ▶ If  $D$  is trivial, let  $D' = D$ .
  - ▶ Otherwise let  $D = (A, D_1, \dots, D_k)$ .
  - ▶ Let  $[D'_1, \dots, D'_i]$  be obtained by
    - ▶ Omitting any  $D_i$  s.t.  $\text{frontier}(D_i) = \epsilon$ .
    - ▶ Applying the IH to any  $D_i$  s.t.  $\text{frontier}(D_i) \neq \epsilon$  in order to obtain a derivation tree in  $G'$ .
  - ▶ Since  $\text{frontier}(D) \neq \epsilon$ , the list  $[D'_1, \dots, D'_i]$  is not empty.
  - ▶ For the  $D_i$  omitted we have that  $\text{label}(D_i) \Rightarrow_G^* \epsilon$ , so  $\text{label}(D_i)$  was nullable.
  - ▶ Therefore  $\text{label}(D'_1) \cdots \text{label}(D'_i)$  is obtained from  $\text{label}(D_1) \cdots \text{label}(D_k)$  by omitting some nullable nonterminals.
  - ▶  $A \rightarrow \text{label}(D_1) \cdots \text{label}(D_k)$  was a production of  $G$ , and therefore  $A \rightarrow \text{label}(D'_1) \cdots \text{label}(D'_i)$  is a production of  $G'$ .

# Equivalence of $G$ and $G'$

- ▶ Therefore  $D' := \text{tree}(A, D'_1, \dots, D'_l)$  is a derivation tree of  $G'$  and  $\text{frontier}(D) = \text{frontier}(D')$ .

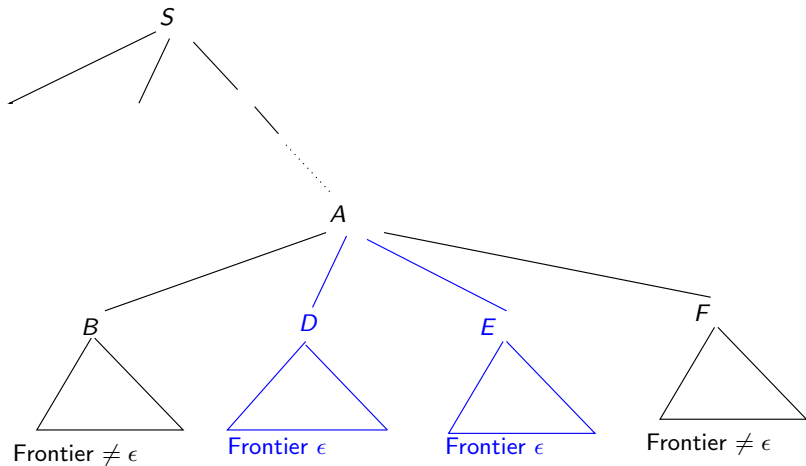
▶ Now

$$\begin{aligned}
 L(G) \setminus \{\epsilon\} &= \{w \in T^* \setminus \epsilon \mid S \Rightarrow_G w\} \\
 &\subseteq \{w \in T^* \mid S \Rightarrow_{G'} w\} \\
 &= L(G')
 \end{aligned}$$

- ▶ The modification done can be described as follows:
- ▶ Assume a derivation tree  $D$  in  $G$  with root  $A$  and frontier  $\neq \epsilon$ .
- ▶ Take any subtree with frontier  $\epsilon$ , which is not contained in a larger subtree with same frontier  $\epsilon$ .
  - ▶ So any largest subtree with frontier  $\epsilon$ .
- ▶ The root of subtree is nullable, since from it we derive  $\epsilon$ .
- ▶ Omit this subtree with its root.
- ▶ The result of removing all such subtrees is a derivation tree in  $D'$ .

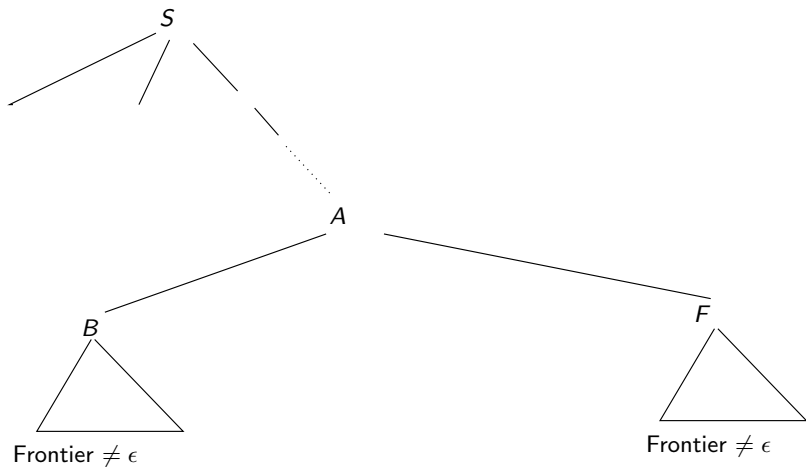
# Picture

The blue subtrees will be removed:



# Picture

After removing the subtrees:



## Step 2: Removal of Silent Productions

### Lemma

*Assume  $G$  a CFG. Then there exist a CFG  $G'$  with no silent productions, i.e. no productions of the form  $A \rightarrow A'$  for nonterminals  $A, A'$  s.t.*

$$L(G) = L(G')$$

*If  $G$  has no null productions,  $G'$  can be chosen to have no null productions as well.*

## Proof

- ▶ Let  $G = (T, N, S, P)$ .
- ▶ Let  $G = (T, N, S, P')$  where  $P'$  consists of all productions  $A \rightarrow w$  where  $w \neq A$  and

$$A \Rightarrow_G^* B \rightarrow w$$

- ▶  $L(G') \subseteq L(G)$ , since any derivation step  $uAv \Rightarrow_{G'} uwv$  where  $A \rightarrow_G w$  is as above can be replaced by  $uAv \Rightarrow_G^* uBv \Rightarrow_G uwv$ .

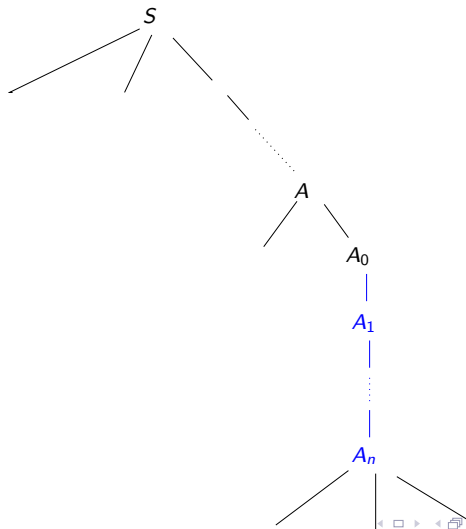
# Proof of $L(G) \subseteq L(G')$

- ▶ We prove that if  $D$  is a derivation tree in  $G$  with frontier  $w \in T^*$ , then there exists a derivation tree  $D'$  in  $G'$  with the same root and frontier.
- ▶ Intuitively  $D'$  is obtained by contracting in  $D$  any chains of silent productions  $A \rightarrow A_1 \rightarrow A_2 \rightarrow \cdots \rightarrow A_n$  where  $A_n$  is the head of a non-trivial derivation tree to  $A$ .



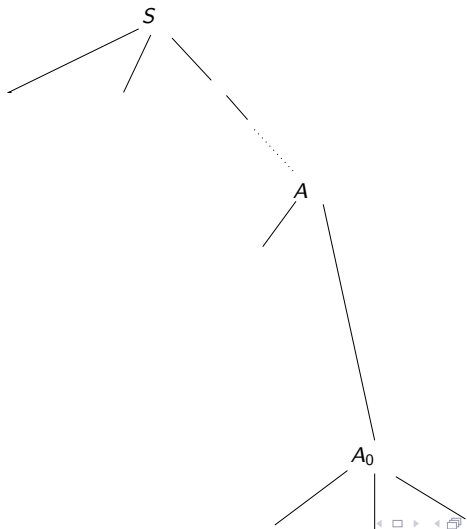
# Picture

The blue chain of silent productions will be removed:



# Picture

After removing the subtrees:



# Removing Silent Productions

- ▶ Formally this can be done by induction on the derivation trees.
- ▶ Let  $D$  be a derivation tree with frontier  $w$  and root  $A$ .
- ▶ If  $D = \text{tree}(A)$ , Let  $D' = D$ .
- ▶ If  $D = \text{tree}(A, D')$  with  $\text{label}(D') \in A$ , let  $D' = \text{tree}(B, D_1, \dots, D_n)$  be the derivation obtained from  $D'$  by IH.
  - ▶ Then we have

$$A \Rightarrow_G B \Rightarrow_G^* C \longrightarrow \text{label}(D_1) \cdots \text{label}(D_k)$$

so

$$A \longrightarrow \text{label}(D_1) \cdots \text{label}(D_k)$$

is a production of  $G'$ .

- ▶ Therefore  $\text{tree}(A, D_1, \dots, D_n)$  is a derivation as desired.

# Removing Silent Productions

- ▶ Otherwise  $D = \text{tree}(A, D_1, \dots, D_n)$  with  $n \geq 2$  or  $\text{label}(D_1) \in T$ .
  - ▶ Let  $D'_i$  a derivation of  $\text{frontier}(D_i)$  with  $\text{label}(D'_i) = \text{label}(D_i)$  obtained by the IH from  $D_i$ .
  - ▶ Then  $\text{tree}(A, D'_1, \dots, D'_n)$  is a derivation as desired.

# Step 3

## Lemma

Let  $G = (T, N, S, P)$  be a CFG. Then we can obtain a grammar  $G'$  s.t.

- ▶ Productions in  $G'$  are either of the form  $A \rightarrow a$  for  $A \in N$  and  $a \in T$ , or of the form  $A \rightarrow w$  where  $w \in N^*$ , so  $w$  consists only of nonterminals.
- ▶  $L(G') = L(G)$ .
- ▶ If  $G$  has no null or silent productions, so does  $G'$ .

# Proof

Let  $G = (T, N, S, P)$ .

- ▶ Let  $G' = (T, N', S, P')$  where
- ▶  $N'$  is obtained by adding to  $N$  for each  $a \in T$  a new symbol  $T_a$ .
- ▶  $P'$  contains
  - ▶ Productions  $T_a \rightarrow a$  for  $a \in T$ ,
  - ▶ For any production  $T \rightarrow w$  a production  $T \rightarrow w'$  where  $w'$  is obtained from  $w$  by replacing each terminal  $a$  by  $T_a$ .
- ▶ It is easy to see that  $L(G') = L(G)$ .

# Proof of Chomsky Normal Form Theorem

- ▶ We prove now the Chomsky Normal Form Theorem.
- ▶ By step 1 - 3 there exists a grammar  $G' = (T, N', S', P')$  s.t.
  - ▶  $L(G') = L(G) \setminus \{\epsilon\}$ ,
  - ▶  $G'$  has no null or silent productions,
  - ▶ the productions of  $G'$  are either  $A \rightarrow a$  or  $A \rightarrow w$  with  $w \in N^*$ .
- ▶ Let  $G'' = (T, N'', S', P'')$  where
  - ▶  $P''$  is obtained by replacing a production  $A \rightarrow B_1 B_2 \cdots B_n$  with  $n \geq 3$  by

$$A \rightarrow B_1 C_1, \quad C_1 \rightarrow B_2 C_2, \quad \dots, \quad C_{n-3} \rightarrow B_{n-3} C_{n-2}, \\ C_{n-2} \rightarrow B_{n-1} B_n$$

for new symbols  $C_1, \dots, C_{n-2}$  (where for each production different symbols are added).

- ▶  $N''$  is obtained from  $N$  by adding all new symbols  $C_i$ .

# Proof of Chomsky Normal Form Theorem

- ▶  $G''$  is in Chomsky Normal Form.
- ▶ Obviously  $L(G') \subseteq L(G'')$ . since derivation steps

$$vAw \Rightarrow vB_1 \cdots B_n w$$

using production  $A \rightarrow B_1 \cdots B_n$  as above can be replaced by

$$\begin{aligned} vAw &\Rightarrow vB_1 C_1 w \Rightarrow vB_1 B_2 C_2 w \Rightarrow \cdots \Rightarrow vB_1 B_2 \cdots B_{n-3} C_{n-2} w \\ &\Rightarrow vB_1 B_2 \cdots B_{n-3} B_{n-2} B_{n-1} w \end{aligned}$$



# Proof of Chomsky Normal Form Theorem

- ▶ On the otherhand, let for  $w \in (T \cup N'')$   $\hat{w}$  be defined as the result of replacing occurrences of
  - ▶  $C_1$  by  $B_2B_3 \cdots B_n$ ,
  - ▶  $C_2$  by  $B_3B_3 \cdots B_n$ ,
  - ▶ etc.
  - ▶  $C_{n-2}$  by  $B_{n-1}B_n$ .
- ▶ Then for any production  $D \rightarrow_{G''} w$  of  $G''$  we have either  $D \rightarrow_{G'} \hat{w}$  or  $D = \hat{w}$ .
- ▶ Therefore, if  $S' \Rightarrow_{G''}^* w$ , then  $S' \Rightarrow_{G'}^* \hat{w}$ .
- ▶ Therefore using that for  $w \in T^*$  we have  $w = \hat{w}$ :

$$\begin{aligned}
 L(G'') &= \{w \in T^* \mid S' \Rightarrow_{G''}^* w\} \\
 &= \{\hat{w} \mid w \in T^* \wedge S' \Rightarrow_{G''}^* w\} \\
 &\subseteq \{\hat{w} \mid w \in T^* \wedge S' \Rightarrow_{G'}^* \hat{w}\} \\
 &= \{w \in T^* \mid S' \Rightarrow_{G'}^* w\} \\
 &= L(G')
 \end{aligned}$$

# A recognition Algorithm for CFG (14.3.2)

- ▶ We present an algorithm for deciding for a CFG  $G$  in Chomsky Normal Form and a string  $w$  whether  $w \in L(G)$ .
- ▶ This algorithm is called the Cocke-Younger-Kasami (CYK) algorithm.

# Cocke-Younger-Kasami Algorithm

- ▶ Let  $G = (T, N, S, P)$ .
- ▶ For a word  $w = t_1 \cdots t_n$  let for  $1 \leq i \leq j \leq n$   $w_{i,k}$  be the subword starting from  $t_i$  of length  $k$ , i.e.

$$w_{i,k} = t_i t_{i+1} \cdots t_{i+k-1}$$

- ▶ We decide more generally for any  $1 \leq i \leq n$ ,  $1 \leq k \leq n - i + 1$  and  $A \in N$  whether  $T \Rightarrow^* w_{i,j}$ .
- ▶ Let

$$N_{i,j} := \{A \in N \mid A \Rightarrow^* w_{i,j}\}$$

# Cocke-Younger-Kasami Algorithm

► We have

- $N_{i,1} = \{A \in N \mid A \rightarrow w_{i,1}\}$ , since a derivation  $A \Rightarrow w_{i,1}$  cannot start with  $A \Rightarrow BC$  since otherwise we would have  $1 = |w_{i,1}| \geq |BC| = 2$ .
- We have

$$N_{i,j+1} = \{A \mid \exists A \rightarrow BC \in P. \exists k < j + 1. 1 \leq k \wedge B \in N_{i,k} \wedge C \in N_{i+k,j+1-k}\}$$

A derivation tree of  $w_{i,j}$  must start with  $A \rightarrow BC$ . Then  $B, C$  derive two subwords of  $w_{i,j}$  which together form  $w_{i,j}$ , both of which are non-empty, so  $B \in N_{i,k}, C \in N_{i+k,j+1-k}$  for some  $1 \leq k \leq j$  as above.

# Cocke-Younger-Kasami Algorithm

- So we can define  $N_{i,j}$  by the following algorithm:

**for**  $i := 1$  **to**  $n$  **do begin**

$N_{i,1} := \{A \mid A \rightarrow w_{i,1} \in P; \}$

**end**

**for**  $j := 2$  **to**  $n$  **do begin**

**for**  $i := 1$  **to**  $n + 1 - j$  **do begin**

$N_{i,j} := \emptyset;$

**for**  $k := 1$  **to**  $j - 1$  **do begin**

$$N_{i,j} := N_{i,j} \cup \{A \mid \exists B, C. A \rightarrow BC \in P$$

$$\wedge B \in N_{i,k} \wedge C \in N_{i+k,j-k}\}$$

**end**

**end**

**end**

# Cocke-Younger-Kasami Algorithm

- ▶ Now we have for  $w \in T^*$

$$\begin{aligned}w \in L(G) &\Leftrightarrow S \Rightarrow^* w \\ &\Leftrightarrow S \in N_{1,|w|}\end{aligned}$$

- ▶ The above algorithm runs in cubic time in  $|w|$ .

II.5.1. Derivation Trees for Context-Free Grammars (14.1)

II.5.2. Uniqueness of Derivation Trees (14.1)

II.5.3. Normal Forms for Context-Free Grammars (14.2)

**II.5.4. The Pumping Lemma for CFG (14.4)**

II.5.5. Floyd's Theorem

# Pumping Lemma for CFG (Repetition from Main Slides)

## Theorem

Let  $L$  be a context free language.. Then there exists a constant  $k$  s.t. for all strings  $z$  of  $L$  s.t.  $|z| \geq k$  there exist  $u, v, w, x, y$  s.t.

- ▶  $z = uvwxy$ ,
- ▶  $|vwx| \leq k$ , i.e. the middle portion is not too long,
- ▶  $|vx| \geq 1$ , i.e.  $v$  or  $x$  are not  $\epsilon$ ,
- ▶  $\forall i \geq 0. uv^i wx^i y \in L$ .



# Proof of the Pumping Lemma for CFG

- ▶ For simplicity we consider CFGs in Chomsky-Normal-Form.

## Lemma (Height of Derivation)

In order to prove the pumping lemma, we need to prove some lemmas.

The first one relates the height of a derivation to the length of the derived string:

### Lemma

*Let  $G = (N, V, S, P)$  be a CFG in Chomsky Normal Form,  $D$  a derivation tree of  $w \in T^*$  of height  $n \geq 1$ . Then  $|w| \leq 2^{n-2}$ .*

# Proof of Lemma (Height of Derivation)

Proof by induction on  $n$ .

- ▶ If  $n = 2$ , then  $D = \text{tree}(A, \text{tree}(w))$ ,  $w \in T$ .  $|w| = 1 = 2^{n-2}$ .
- ▶  $2 \leq n \rightarrow n + 1$ : The rule used at the root must have been  $A \rightarrow BC$ , since, if we had used  $A \rightarrow a$ , we would get a tree of height 2. Let  $D = \text{tree}(A, D_1, D_2)$ . Then

$$\begin{aligned}
 |w| &= |\text{frontier}(w_1).\text{frontier}(w_2)| \\
 &= |\text{frontier}(w_1)| + |\text{frontier}(w_2)| \\
 &\leq 2^{n-2} + 2^{n-2} = 2^{n+1-2}
 \end{aligned}$$

# Definition of Subderivations

## Definition

Let  $G = (T, N, S, P)$  be a grammar,  $D'$ ,  $D$  derivations in  $G$ .

We define what it means for  $D'$  to be a subderivation of  $D$ :

- ▶  $D$  is a subderivation of itself.
- ▶ If  $D''$  is a child of  $D$ , and  $D'$  is a subderivation of  $D''$  then  $D'$  is a subderivation of  $D$ .

$D'$  is a proper subderivation of  $D$  if  $D'$  is a subderivation of  $D$ , but  $D' \neq D$ .

# Existence of Subderivations of smaller heights

## Lemma

*Let  $D$  be a derivation of height  $n$ ,  $1 \leq k < n$ . Then there exist a proper subderivation  $D'$  of  $D$  of height  $k$ .*

# Proof of Lemma (Existence of Subderivations)

- ▶ The proof is by induction on the height  $n$  of  $D$ :
- ▶ In case  $n = 1$ , no  $1 \leq k < n$  exists.
- ▶ In case  $n = 2$ ,  $D = \text{tree}(A, D_1, \dots, D_l)$ . We must have  $l \geq 1$ , otherwise  $\text{height}(D) = 1$ .  $\text{height}(D_1) = 1 = k$ , let  $D' = D_1$ .
- ▶ Induction step  $n \rightarrow n + 1$ , assuming  $n \geq 2$ .
  - ▶ Let  $D = \text{tree}(A, D_1, \dots, D_l)$ .
  - ▶  $n + 1 = \text{height}(D) = \max\{\text{height}(D_1), \dots, \text{height}(D_l)\} + 1$ ,
  - ▶ So there exists an  $i$  s.t.  $\text{height}(D_i) = n$ .
  - ▶ If  $k = n$ , choose  $D' := D_i$ .
  - ▶ If  $k < n$ , then by IH there exist a proper subderivation  $D''$  of  $D_i$  of height  $k$ , which is a proper subderivation of  $D$  as well.  $D' := D''$ .

# Derivations Deriving Non-Empty Strings

## Lemma

*Assume  $G = (T, N, S, P)$  is a CFG with no null-productions. Then if  $A \Rightarrow^* w$ , then  $|w| \geq 1$ .*

**Proof:** If  $v \Rightarrow v'$ , then  $|v'| \geq |v|$ . Therefore if  $v \Rightarrow^* v'$ , then  $|v'| \geq |v|$ .  
Now  $|w| \geq |A| = 1$ .

# Cutting out Subderivations

## Lemma

*Let  $G$  be a CFG.*

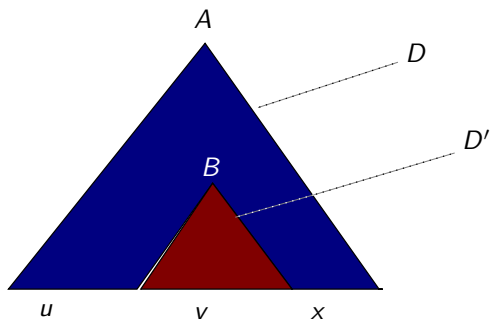
*Let  $D$  be a derivation tree with label  $A$  and frontier  $w$ . Let  $D'$  be a proper subderivation  $D'$  of  $D$  with label  $B$  and frontier  $v$ .*

*Then there exist  $u, x$  s.t.  $w = uvx$ , and a derivation  $A \Rightarrow^* uBx$ .*

*Furthermore, if  $\text{height}(D) \geq 2$  and  $D$  has no null or silent productions, then  $|u| + |x| \geq 1$ .*

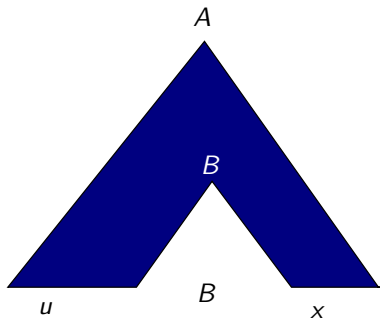


## Picture



# Picture

After cutting out the derivation:



## Proof (Cutting out Subderivations)

Induction on  $D'$  being a subderivation of  $D$ .

We write “the special case” for the condition  
“ $\text{height}(D) \geq 2$  and  $D$  has no null or silent productions”.

Let  $D = \text{tree}(A, D_1, \dots, D_n)$ .

In the special case we get  $n \geq 2$ .

- ▶ (If  $n = 1$ , we would obtain  $D_1 = \text{tree}(a, D'_1, \dots, D'_l)$ ,  $a$  must be a nonterminal, since  $G$  has no silent productions, and therefore  $l = 0$ . But then  $\text{height}(D) = 1$ ).

# Proof (Cutting out Subderivations)

**Case 1:**  $D'$  is a child of  $D$ .

Then  $D = \text{tree}(A, D_1, \dots, D_n)$ ,  $D' = D_i$  some  $i$ .

Let

$$u := \text{frontier}(D_1).\text{frontier}(D_2).\dots.\text{frontier}(D_{i-1})$$

$$x := \text{frontier}(D_{i+1}).\text{frontier}(D_{i+2}).\dots.\text{frontier}(D_n) .$$

Then  $w = uvx$ . Let

$$D'' := \text{tree}(A, D_1, \dots, D_{i-1}, \text{tree}(B), D_{i+1}, \dots, D_n) .$$

$D''$  is a derivation tree of  $uBx$  with the label  $A$ .

Furthermore, in the special case we have  $n \geq 2$ ,  $\text{frontier}(D_i) \neq \epsilon$ , therefore  $u \neq \epsilon$  or  $x \neq \epsilon$ .

## Proof (Cutting out Subderivations)

**Case 1:**  $D'$  is a proper subderivation of a child  $D_i$  of  $D$ . Let  $C$  be the label of  $D_i$ .

Let  $w' := \text{frontier}(D_i)$ . By IH there exists  $u', x'$  s.t.  $w' = u'vx'$ , and a derivation tree  $D'$  with label  $C$  and frontier  $u'Bx'$ .

$$\begin{aligned} u'' &:= \text{frontier}(D_1).\text{frontier}(D_2).\cdots.\text{frontier}(D_{i-1}) \\ x'' &:= \text{frontier}(D_{i+1}).\text{frontier}(D_{i+2}).\cdots.\text{frontier}(D_n) . \end{aligned}$$

Then  $w = u''w'x'' = (u''u')v(x'x'')$ . Let  $u := u''u'$ ,  $x := x'x''$ . Let

$$D'' := \text{tree}(A, D_1, \dots, D_{i-1}, D', D_{i+1}, \dots, D_n) .$$

$D''$  is a derivation tree of  $u''u'Bx'x'' = uBx$ .

Furthermore, in the special case we have as before  $u'' \neq \epsilon$  or  $x'' \neq \epsilon$ , therefore  $u \neq \epsilon$  or  $x \neq \epsilon$ .

# Chains of Derivations

## Definition

Let  $G$  be a CFG. A chain of derivations is a list of derivations  $[D_1, \dots, D_n]$  s.t.  $D_{i+1}$  is a child of  $D_i$  and  $\text{height}(D_n) = 1$ .  $n$  is called the length of the chain of derivations.

# Existence of Chains of Derivations

## Lemma

*Let  $G$  be a CFG. If  $D$  has height  $n$  then there exists a chain of derivations  $[D_1, \dots, D_n]$  of length  $n$  s.t.  $D_1 = D$ .*

# Proof (Existence of Chains of Derivations)

Induction on  $n$ : If  $n = 1$ , then  $[D]$  is a chain of derivations as desired.

$n \rightarrow n + 1$ : Let  $D$  have height  $n + 1$ .

There exist a child  $D_2$  of  $D_1 := D$  which has height  $n$ .

By IH there exist a chain of derivations  $[D_2, D_3, \dots, D_{n+1}]$  of length  $n$  starting with  $D_2$ .

Then  $[D_1, \dots, D_{n+1}]$  is a chain of derivations as desired.



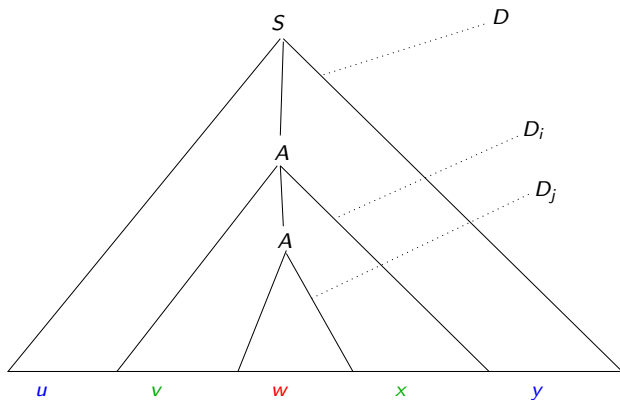
# Proof of the Pumping Lemma

- ▶ Let  $L = L(G)$ . W.l.o.g.  $G$  is in Chomsky Normal Form.
  - ▶ There exist a grammar  $G'$  in Chomsky Normal Form s.t.  $L(G') = L \setminus \{\epsilon\}$ .
  - ▶ If the lemma holds for  $L(G')$ , then it holds for  $L$  as well (with the same constant  $n$ ).
- ▶ Let  $G = (T, N, S, P)$ , and assume  $|N| = l$ .
- ▶ Let  $k := 2^l$ .

# Proof of the Pumping Lemma

- ▶ Assume  $z \in L(G)$ ,  $|z| \geq k = 2^l$ .
- ▶ Let  $D$  be a derivation tree for  $z$ .
- ▶  $\text{height}(D) \geq l + 2$ .
- ▶ There exist a subderivation  $D'$  of  $D$  of height  $l + 2$ .
- ▶ Then there exist a chain of derivations  $[D_1, D_2, \dots, D_{l+2}]$ , s.t.  $D' = D_1$ .
- ▶  $\text{label}(D_i) \in N$  for  $i = 1, \dots, l + 1$ .
- ▶ Therefore there exists  $1 \leq i < j \leq l + 1$  s.t.  $\text{label}(D_i) = \text{label}(D_j) =: A$ .
- ▶ So we have found a subderivation  $D_i$  of  $D$  of height  $\leq l + 2$  which contains a proper subderivation  $D_j$  with the same label.

## Picture



# Proof of the Pumping Lemma

- ▶ Let  $w = \text{frontier}(D_j)$ .
- ▶ By the “Cutting out of Derivations” lemma applied to  $D_i$  and  $D_j$ , we have  $\text{frontier}(D_i) = vwx$  s.t.  $vx \neq \epsilon$ , and  $A \Rightarrow^* vAx$ . (The green derivation on the next picture).
- ▶ By the “Cutting out of Derivations” lemma applied to  $D$  and  $D_i$  we have  $z = uvwxy$  for some strings  $u, y$ , and we have  $S \Rightarrow^* uAy$ . (The blue derivation on the next picture).
- ▶ Furthermore by  $D_j$  we have  $A \Rightarrow^* w$ . (The red derivation on the next picture).
- ▶  $\text{height}(D_i) \leq l + 2$ , therefore  $|vwx| \leq 2^l = k$ .
- ▶ We obtain

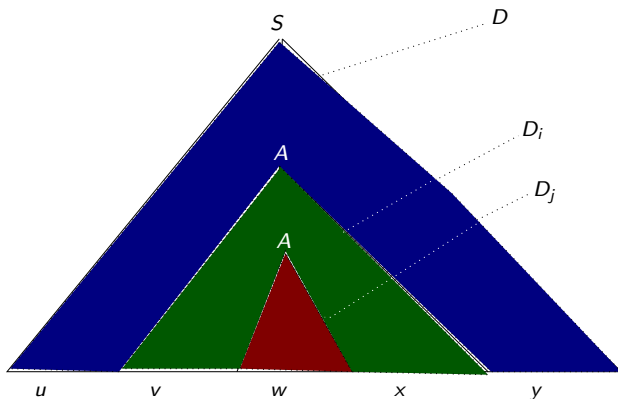
$$A \Rightarrow^* vAx \Rightarrow^* v^2Ax^2 \Rightarrow^* \dots$$

$$\forall i \geq 0. A \Rightarrow^* v^iAx^i$$

$$\forall i \geq 0. S \Rightarrow^* uAy \Rightarrow^* uv^iAx^i y \Rightarrow^* uv^iwx^i y$$

$$\forall i \geq 0. uv^iwx^i y \in L(G)$$

## Picture



II.5.1. Derivation Trees for Context-Free Grammars (14.1)

II.5.2. Uniqueness of Derivation Trees (14.1)

II.5.3. Normal Forms for Context-Free Grammars (14.2)

II.5.4. The Pumping Lemma for CFG (14.4)

II.5.5. Floyd's Theorem

# Floyd's Theorem

- ▶ We are going to show Floyd's Theorem, that a programming language with variable declarations cannot be defined in a context free way.
- ▶ We need some restrictions on what is meant by a programming language.
- ▶ In order to illustrate it, we consider a restricted form of the while language already considered.

# Grammar for While Programs

Consider the grammar studied in 2.1.3:

<b>grammar</b>	$G^{while}$
<b>import</b>	$G^{Arithmetic\_Expression}, G^{Boolean\_Expression}$
<b>terminals</b>	<b>skip, if, then, else, fi, while, do, od, :=, ;</b>
<b>nonterminals</b>	<i>Program</i>
<b>start symbol</b>	<i>Program</i>
<b>productions</b>	$Program \rightarrow \mathbf{skip}$ $Program \rightarrow Id := AExp$ $Program \rightarrow Program ; Program$ $Program \rightarrow \mathbf{if} BExp \mathbf{then} Program \mathbf{else} Program \mathbf{fi}$ $Program \rightarrow \mathbf{while} BExp \mathbf{do} Program \mathbf{od}$



# Modified Grammar

We simplify it by adding variable declarations, and omit **if\_then\_else** and **while** and disambiguate the sequencing construct. We obtain the following grammar:

## Grammar for Programs With Declarations

<b>grammar</b>	$G^{Programs\_with\_declarations}$
<b>import</b>	$G^{Arithmetic\_Expression}, G^{Declarations}$
<b>terminals</b>	<b>skip, begin, end, :=, ;</b>
<b>nonterminals</b>	<i>Program, CommandList, Command</i>
<b>start symbol</b>	<i>Program</i>
<b>productions</b>	$Program \rightarrow \mathbf{begin} \textit{Declaration} ; \textit{CommandList} \mathbf{end}$ $CommandList \rightarrow \textit{Command}$ $CommandList \rightarrow \textit{CommandList} ; \textit{Command}$ $Command \rightarrow \mathbf{skip}$ $Command \rightarrow \textit{Id} := \textit{AExp}$

## Declarations

<b>grammar</b>	$G^{Declarations}$
<b>import</b>	$G^{Identifier}$
<b>terminals</b>	<b>nat</b>
<b>nonterminals</b>	<i>Declaration, IdentifierList</i>
<b>start symbol</b>	<i>Declaration</i>
<b>productions</b>	<i>Declaration</i> $\rightarrow$ <b>nat</b> <i>IdentifierList</i> <i>IdentifierList</i> $\rightarrow$ <i>Identifier</i> <i>IdentifierList</i> $\rightarrow$ <i>Identifier, IdentifierList</i>

# Floyd's Theorem II.5.4

## Theorem

Let

$$Decl_{n,m} := \mathbf{begin\ nat\ } a^n b^m \mathbf{ ;\ } a^n b^m := 0 \mathbf{ end}$$

Let

$$Decl_{n,m} \subseteq L$$

be a programming language.

Assume some sanity condition on what it means for  $L$  to be a programming language, including that in  $L$  all identifiers need to be declared before being used.

Then  $L$  is not context free.

# Floyd's Theorem II.5.4

## Theorem (Cont)

*The sanity conditions are the following:*

- 1. In any program of  $L$  all identifiers of the program are declared by strings  $t$  a where  $t$  is a type identifier, which are special elements of the alphabet.*
- 2. **nat** is one type identifier.*
- 3. The keywords and “:=” are special elements of the alphabet.*
- 4. Identifiers are strings.*
- 5. **begin** and **end** need to be balanced in  $L$ .*
- 6. For an identifier  $s$   $s := 0$  is an instruction using identifier  $s$ .*
- 7. Removing of one or more of the symbols :=, 0 from  $s := 0$  yields a string which is not an instruction.*
- 8. Programs in  $L$  contain at least one instruction.*

# Proof (Floyd's Theorem)

- ▶ Assume  $L$  is context free and let  $k$  be the constant of the pumping lemma for CFG.
- ▶ Let  $n, m > k$ .
- ▶  $Decl_{n,m} \in L$  and  $|Decl_{n,m}| > k$ .
- ▶ By the pumping lemma there exists  $u, v, w, x, y$  s.t.

$$Decl_{n,m} = uvwxy \wedge vx \neq \epsilon \wedge |vwx| \leq k \wedge \forall i \geq 0. uv^i wx^i y \in L$$

- ▶ Let  $P_i := uv^i wx^i y$ .
- ▶ So

$$uvwxy = Decl_{n,m} = \mathbf{begin\ nat\ } a^n b^m ; a^n b^m := 0 \mathbf{end}$$

# Proof (Floyd's Theorem)

$uvwxy = Decl_{n,n} = \mathbf{begin\ nat\ } a^n b^m ; a^n b^m := 0 \mathbf{end}$

$P_i := uv^i wx^i y \in L$

- ▶ Because  $|vwx| \leq k$ ,  $vwx$  cannot contain both **begin** and **end**.
- ▶ If  $v$  or  $x$  contained **begin**, or **end**, then  $P_0$  would contain only one of these two keywords, contradicting the fact that **begin** and **end** need to be balanced.
- ▶ If  $v$  contains **nat**, then  $vwx$  must be part of **nat**  $a^n b^n$ .  
Then  $P_0$  wouldn't contain a declaration of  $a^n b^m$  which is used in the statement  $a^n b^m := 0$  which is part of  $P_0$ .
- ▶ Therefore we have that  $vwx$  must be part of  $a^n b^m ; a^n b^m := 0$ .

# Proof (Floyd's Theorem)

$uvwxy = Decll_{n,n} = \mathbf{begin\ nat\ } a^n b^m ; a^n b^m := 0 \mathbf{\ end}$

$P_i := uv^i wx^i y \in L$

$vwx$  part of  $a^n b^m ; a^n b^m := 0$

- ▶ If  $:= 0$  would overlap with  $x$ , then  $P_0$  would not have one or more of those symbols, and the part after  $;$  wouldn't be an instruction.
- ▶ So  $vwx$  is part of  $a^n b^m ; a^n b^m$ .



# Proof (Floyd's Theorem)

$uvwxy = Decl_{n,n} = \mathbf{begin\ nat\ } a^n b^m ; a^n b^m := 0 \mathbf{end}$

$P_i := uv^i wx^i y \in L$

$vwx$  part of  $a^n b^m ; a^n b^m$

- ▶ If  $v$  or  $x$  contain  $;$  then  $P_0$  doesn't contain a  $;$ , so it consists only of a declaration and has no instruction.
- ▶ If  $w$  contains  $;$ , then  $v$  is contained in  $b^m$ ,  $x$  contained in  $a^n$ , therefore in  $P_0$  the declared identifier and used identifier are different, contradicting that identifiers need to be declared.
- ▶ So  $vwx$  is part of  $a^n b^m$  before or after the  $;$ . But then in  $P_0$  the identifiers before and after  $;$  are different, contradicting that identifiers need to be declared.