# CSCI 742 - Compiler Construction

Lecture 17

Chomsky Normal Form (CNF)

Instructor: Hossein Hojjat

February 23, 2018

# Directionality

## Directional Methods

- Process the input symbol by symbol from Left to right
- Advantage: parsing starts and makes progress before the last symbol of the input is seen
- Example: LL and LR parsers

## Non-directional Methods

- Allow access to input in an arbitrary order
- Require the entire input to be in memory before parsing can start
- Advantage: allow more flexible grammars than directional parsers
- Example: CYK parser

# Directionality

### Directional Methods

- Process the input symbol by symbol from Left to right
- Advantage: parsing starts and makes progress before the last symbol of the input is seen
- Example: LL and LR parsers

### Non-directional Methods

- Allow access to input in an arbitrary order
- Require the entire input to be in memory before parsing can start
- Advantage: allow more flexible grammars than directional parsers
- Example: CYK parser

# More Powerful Parsers

- LL and LR: deterministic, directional, linear-time recognition of restricted forms of context-free grammars

How can we design algorithms to parse more grammars non-directionally?
(if we allow more time-consuming algorithms)

Some ideas:

- **Naïve:** enumerate everything!
- **Backtracking:** try subtrees and discard partial solutions if unsuccessful
- **Dynamic Programming:** save partial solutions in a table for later use

# CYK Parsing

- CYK recognizes any context-free grammar in Chomsky Normal Form
- Named after J. Cocke, D.H. Younger and T. Kasami
- Uses dynamic programming
- **Bottom-up:** reduces already recognized right-hand side of a production rule to its left-hand side non-terminal
- **Non-directional:** accesses input in arbitrary order so requires the entire input to be in memory before parsing can start

In this lecture we learn about Chomsky Normal Form (CNF)

# Chomsky Normal Form (CNF)

A CFG is in Chomsky Normal Form if each rule is of the form

$$A \to BC$$
$$A \to a$$

where

- $a$ is any terminal
- $A$,$B$,$C$ are non-terminals
- $B$, $C$ cannot be start variable

We allow the rule $S \to \epsilon$ if $\epsilon \in L$

# Conversion to Chomsky Normal Form (CNF)

**Steps:** (not in the optimal order)

1. remove unproductive non-terminals
2. remove unreachable non-terminals
3. remove $\epsilon$-production rules $X \to \epsilon$ ($X$ is not start non-terminal)
4. remove single non-terminal productions (unit production rules) ($X \to Y$)
5. reduce arity of every production to less than two
6. make terminals occur alone on right-hand side

# (1) Unproductive Non-terminals

- Consider the following grammar with start non-terminal “stmt”

$$
\begin{aligned}
\text{stmt} &\rightarrow \text{identifier} := \text{identifier} \\
&\mid \ \texttt{while}\ (\text{expr})\ \text{stmt} \\
&\mid \ \texttt{if}\ (\text{expr})\ \text{stmt}\ \texttt{else}\ \text{stmt} \\
\text{expr} &\rightarrow \text{term} + \text{term} \mid \text{term} - \text{term} \\
\text{term} &\rightarrow \text{factor} * \text{factor} \\
\text{factor} &\rightarrow (\text{expr})
\end{aligned}
$$

- There is no derivation of a sequence of tokens from expr
- Every derivation step of expr has at least one expr, term, or factor
- If a non-terminal cannot derive sequence of tokens we call it **unproductive**

# (1) Unproductive Non-terminals

**Productive Non-terminals**

- Productive non-terminals are obtained using these two rules (what remains is unproductive)

1) Terminals are productive
2) If $A \rightarrow \alpha$ is a production rule and each non-terminal symbols of $\alpha$ is productive then $A$ is also productive
   ($\alpha$ can also be $\epsilon$)

**Remove Unproductive Non-terminals**

- Remove all production rules in which an unproductive non-terminal appears either on the left or the right

# Exercise

**Question:**

- Remove all the unproductive non-terminals from the following grammar.

$$
\begin{aligned}
S &\to B \mid AC \\
B &\to aAa \\
A &\to \epsilon \\
C &\to cC \mid DA \\
D &\to C
\end{aligned}
$$

# Exercise

**Question:**

- Remove all the unproductive non-terminals from the following grammar.

$$
\begin{aligned}
S &\rightarrow B \mid AC \\
B &\rightarrow aAa \\
A &\rightarrow \epsilon \\
C &\rightarrow cC \mid DA \\
D &\rightarrow C
\end{aligned}
$$

**Answer:**

$$
\begin{aligned}
S &\rightarrow B \\
B &\rightarrow aAa \\
A &\rightarrow \epsilon
\end{aligned}
$$

# (2) Unreachable non-terminals

- Consider the following grammar with start non-terminal "program"

$$
\begin{aligned}
\text{program} &\rightarrow \text{stmt} \mid \text{stmt program} \\
\text{stmt} &\rightarrow \text{assignment} \mid \text{whileStmt} \\
\text{assignment} &\rightarrow \text{expr} = \text{expr} \\
\text{ifStmt} &\rightarrow \texttt{if}\ (\text{expr})\ \text{stmt}\ \texttt{else}\ \text{stmt} \\
\text{whileStmt} &\rightarrow \texttt{while}\ (\text{expr})\ \text{stmt}
\end{aligned}
$$

- No way to reach non-terminal "ifStmt" from "program"

# (2) Unreachable non-terminals

**Reachable Non-terminals**

- Reachable non-terminals are obtained using these two rules (what remains is unreachable)

1) Starting non-terminal is reachable
2) If $A \rightarrow \alpha$ is a production rule and $A$ is reachable, each non-terminal symbols of $\alpha$ is also reachable

**Remove Unreachable Non-terminals**

- Remove all production rules in which an unreachable non-terminal appears either on the left or the right

# (3) Removing $\epsilon$-Production Rules

- Ensure only top-level non-terminal can be nullable

| Original Grammar | Grammar after removing $\epsilon$-rules |
|---|---|
| program $\rightarrow$ stmtSeq | program $\rightarrow$ "" $\mid$ stmtSeq |
| stmtSeq $\rightarrow$ stmt $\mid$ stmt **;** stmtSeq | stmtSeq $\rightarrow$ stmt $\mid$ stmt **;** stmtSeq |
| stmt $\rightarrow$ "" $\mid$ assignment | $\mid$ **;** stmtSeq $\mid$ stmt **;** $\mid$ **;** |
| $\mid$ whileStmt $\mid$ blockStmt | stmt $\rightarrow$ assignment |
| blockStmt $\rightarrow$ **{** stmtSeq **}** | $\mid$ whileStmt $\mid$ blockStmt |
| assignment $\rightarrow$ expr **=** expr | blockStmt $\rightarrow$ **{** stmtSeq **}** $\mid$ **{}** |
| whileStmt $\rightarrow$ `while` **(**expr**)** stmt | assignment $\rightarrow$ expr **=** expr |
| expr $\rightarrow$ identifier | whileStmt $\rightarrow$ `while` **(**expr**)** stmt |
| | whileStmt $\rightarrow$ `while` **(**expr**)** |
| | expr $\rightarrow$ identifier |

# Recap: Nullable Non-terminals

- **Definition:** Variable $X$ is nullable if $X \Rightarrow^* \epsilon$

- Rules to compute the nullable variables of a grammar:

1) If $A \rightarrow \epsilon$ is a production rule then $A$ is *nullable*
2) If $B \rightarrow X_1 X_2 \cdots X_n$ is a production rule and all the $X_i$ are *nullable* then B is also *nullable*

# (3) Removing $\epsilon$-Production Rules

- Compute the set of *nullable* non-terminals
- For each rule $A \to X_1 \cdots X_n$ add **all** production rules that that can be formed by eliminating **any** combination of *nullable* $X_i$'s
- Repeat the above step for the newly added rules
- Remove all rules with empty right-hand sides
- If starting symbol $S$ was nullable, then introduce a new start symbol $S'$ instead, and add rule $S' \to S \mid$ ""
- Note: number of added rules for $A \to X_1 \cdots X_n$ is $O(2^k)$ (where $k$ is the number of *nullable* $X_i$'s)

# (3) Removing $\epsilon$-Production Rules

- Since stmtSeq is *nullable*, the rule
  blockStmt $\rightarrow$ **{** stmtSeq **}**
  gives
  blockStmt $\rightarrow$ **{** stmtSeq **}** $|$ **{ }**
- Since stmtSeq and stmt are *nullable*, the rule
  stmtSeq $\rightarrow$ stmt $|$ stmt **;** stmtSeq
  gives
  stmtSeq $\rightarrow$ stmt $|$ stmt **;** stmtSeq
  $|$ **;** stmtSeq $|$ stmt **;** $|$ **;**

# Exercise

**Question:**

1) Remove the $\epsilon$ production rules from the following grammar.
2) Remove unproductive non-terminals after step 1.

$$S \to ABC$$

$$B \to CAb \mid b$$

$$C \to ASD \mid AD$$

$$D \to CaA \mid \epsilon$$

$$A \to \epsilon$$

# Exercise

**Question:**

1) Remove the $\epsilon$ production rules from the following grammar.
2) Remove unproductive non-terminals after step 1.

$$S \to ABC$$
$$B \to CAb \mid b$$
$$C \to ASD \mid AD$$
$$D \to CaA \mid \epsilon$$
$$A \to \epsilon$$

**Answer:**

After removing $\epsilon$ rules:

$$S \to ABC \mid AB \mid B \mid BC$$
$$B \to CAb \mid Ab \mid b \mid Cb$$
$$C \to ASD \mid AD \mid AS \mid S \mid SD \mid A \mid D$$
$$D \to CaA \mid Ca \mid a \mid aA$$

# (4) Eliminating unit productions

- Single production rule is of the form

$$X \to Y$$

where $X$, $Y$ are non-terminals

$$
\begin{aligned}
\text{program} &\to \text{stmtSeq} \\
\text{stmtSeq} &\to \text{stmt} \\
&\mid \quad \text{stmt } \mathbf{;} \text{ stmtSeq} \\
\text{stmt} &\to \text{assignment} \mid \text{whileStmt} \\
\text{assignment} &\to \text{expr} \mathbf{=} \text{expr} \\
\text{whileStmt} &\to \texttt{while} \; \mathbf{(}\text{expr}\mathbf{)} \text{ stmt}
\end{aligned}
$$

# (4) Eliminating unit productions

- If there is a unit production $X \to Y$ put an edge $(X, Y)$ into graph
- If there is a path from $Y$ to $Z$ in the graph, and there is rule $Z \to X_1 X_2 \cdots X_n$ then add rule $Y \to X_1 X_2 \cdots X_n$

At the end, remove all unit productions

# (4) Eliminating unit productions

$$\begin{aligned}
\text{program} &\to \text{stmtSeq} \\
\text{stmtSeq} &\to \text{stmt} \mid \text{stmt}\ \textbf{;}\ \text{stmtSeq} \\
\text{stmt} &\to \text{assignment} \mid \text{whileStmt} \\
\text{assignment} &\to \text{expr}\ \textbf{=}\ \text{expr} \\
\text{whileStmt} &\to \texttt{while}\ \textbf{(}\text{expr}\textbf{)}\ \text{stmt}
\end{aligned}$$

After removing unit productions:

$$\begin{aligned}
\text{program} &\to \text{expr}\ \textbf{=}\ \text{expr} \mid \texttt{while}\ \textbf{(}\text{expr}\textbf{)}\ \text{stmt} \\
&\mid \quad \text{stmt}\ \textbf{;}\ \text{stmtSeq} \\
\text{stmtSeq} &\to \text{expr}\ \textbf{=}\ \text{expr} \mid \texttt{while}\ \textbf{(}\text{expr}\textbf{)}\ \text{stmt} \\
&\mid \quad \text{stmt}\ \textbf{;}\ \text{stmtSeq} \\
\text{stmt} &\to \text{expr}\ \textbf{=}\ \text{expr} \mid \texttt{while}\ \textbf{(}\text{expr}\textbf{)}\ \text{stmt} \\
\text{assignment} &\to \text{expr}\ \textbf{=}\ \text{expr} \\
\text{whileStmt} &\to \texttt{while}\ \textbf{(}\text{expr}\textbf{)}\ \text{stmt}
\end{aligned}$$

# (5) Reducing Arity

- No more than 2 non-terminals on RHS

$$\text{stmt} \rightarrow \texttt{while}\ \textbf{(}\text{expr}\textbf{)}\ \text{stmt}$$

- becomes

$$\begin{aligned} \text{stmt} &\rightarrow \texttt{while}\ \text{stmt}_1 \\ \text{stmt}_1 &\rightarrow \textbf{(}\text{stmt}_2 \\ \text{stmt}_2 &\rightarrow \text{expr}\ \text{stmt}_3 \\ \text{stmt}_3 &\rightarrow \textbf{)}\text{stmt} \end{aligned}$$

# (6) A non-terminal for each terminal

$$\text{stmt} \rightarrow \texttt{while}\ \mathbf{(}\text{expr}\mathbf{)}\ \text{stmt}$$

- becomes

$$\begin{aligned}
\text{stmt} &\rightarrow N_{\texttt{while}}\ \text{stmt}_1 \\
\text{stmt}_1 &\rightarrow N_{(}\text{stmt}_2 \\
\text{stmt}_2 &\rightarrow \text{expr}\ \text{stmt}_3 \\
\text{stmt}_3 &\rightarrow N_{)}\text{stmt} \\
N_{\texttt{while}} &\rightarrow \texttt{while} \\
N_{(} &\rightarrow ( \\
N_{)} &\rightarrow )
\end{aligned}$$

# Order of steps in conversion to CNF

1. remove unproductive non-terminals (optional)
2. remove unreachable non-terminals (optional)
3. make terminals occur alone on right-hand side
4. reduce arity of every production to $\leq 2$
5. remove epsilons
6. remove unit productions $X \to Y$
7. unproductive non-terminals
8. unreachable non-terminals