Normal Forms and Pushdown Automata

Mridul Aanjaneya

Stanford University

July 19, 2012
• No class on Thursday (07/26).
• Practice problems.
• Office hours on Thursday (07/26) (1PM - 4PM) and (5PM - 8PM) (Gates 104).
• Office hours on Monday (07/30) (6PM - 9PM) (Gates 104).
We can almost avoid using productions of the form $A \rightarrow \epsilon$ (called $\epsilon$-productions).

- The problem is that $\epsilon$ cannot be in the language of any grammar that has no $\epsilon$-productions.

**Theorem**

If $L$ is a CFL, then $L - \{\epsilon\}$ has a CFG with no $\epsilon$-productions.
- To eliminate \( \varepsilon \)-productions, we first need to discover the nullable symbols = variables \( A \) such that \( A \Rightarrow^* \varepsilon \).

- **Basis:** If there is a production \( A \rightarrow \varepsilon \), then \( A \) is nullable.

- **Induction:** If there is a production \( A \rightarrow \alpha \), and all symbols of \( \alpha \) are nullable, then \( A \) is nullable.
Example: Nullable Symbols

S → AB, A → aA | ε, B → bB | A

- **Basis:** A is nullable because of $A \rightarrow \varepsilon$.
- **Induction:** B is nullable because of $B \rightarrow A$.
- Then, S is nullable because of $S \rightarrow AB$. 
Proof of Algorithm: Nullable Symbols

- Proof is very much like that for the algorithm for testing variables that derive terminal strings.
- Left to the imagination!
Eliminating $\varepsilon$-productions

- **Key idea**: turn each production

  \[
  A \rightarrow X_1 \ldots X_n
  \]

  into a family of productions.

- For each subset of nullable $X$’s, there is one production with those eliminated from the right side in advance.
  - Except, if all $X$’s are nullable, do not make a production with $\varepsilon$ as the right hand side.
Example: Eliminating $\varepsilon$-productions

$S \rightarrow ABC, A \rightarrow aA \mid \varepsilon, B \rightarrow bB \mid \varepsilon, C \rightarrow \varepsilon$

- A, B, C and S are all nullable.
- New grammar:

  $S \rightarrow ABC \mid AB \mid AC \mid BC \mid A \mid B \mid C$

  $A \rightarrow aA \mid a$

  $B \rightarrow bB \mid b$

- **Note:** C is now useless, eliminate its productions.
Why It Works

• **Prove** that for all variables A:
  1. If $w \neq \varepsilon$ and $A \Rightarrow_{\text{old}}^* w$, then $A \Rightarrow_{\text{new}}^* w$.
  2. If $A \Rightarrow_{\text{new}}^* w$, then $w \neq \varepsilon$ and $A \Rightarrow_{\text{old}}^* w$.

• Then, letting A be the start symbol proves that $L(\text{new}) = L(\text{old}) - \{\varepsilon\}$.

• (1) is an induction on the number of steps by which A derives $w$ in the old grammar.
Proof of 1 - Basis

- If the old derivation is one step, then $A \rightarrow w$ must be a production.
- Since $w \neq \varepsilon$, this production also appears in the new grammar.
- Thus, $A \Rightarrow_{\text{new}} w$. 
Proof of 1 - Induction

- Let \( A \Rightarrow_{\text{old}}^* w \) be an \( n \)-step derivation, and assume the IH for derivations of less than \( n \) steps.
- Let the first step be \( A \Rightarrow_{\text{old}} X_1 \ldots X_n \).
- Then \( w \) can be broken into \( w = w_1 \ldots w_n \), where \( X_i \Rightarrow_{\text{old}}^* w_i \), for all \( i \), in fewer than \( n \) steps.
Proof of 1 - Induction

- By the IH, if \( w_i \neq \varepsilon \), then \( X_i \Rightarrow^{*}_{\text{new}} w_i \).
- Also, the new grammar has a production with \( A \) on the left, and just those \( X_i \)'s on the right such that \( w_i \neq \varepsilon \).
  - **Note:** They all **cannot** be \( \varepsilon \), because \( w \neq \varepsilon \).
- Follow a use of this production by the derivations \( X_i \Rightarrow^{*}_{\text{new}} w_i \) to show that \( A \) derives \( w \) in the new grammar.
We also need to show part (2) - if \( w \) is derived from \( A \) in the new grammar, then it is also derived in the old.

Induction on number of steps in the derivation.

Left as exercise.
Unit Productions

• A unit production is one whose right hand side consists of exactly one variable.
• These productions can be eliminated.
• Key idea: If $A \Rightarrow^* B$ by a series of unit productions, and $B \rightarrow \alpha$ is a non-unit production, then add the production $A \rightarrow \alpha$.
• Then drop all unit productions.
• Find all pairs \((A, B)\) such that \(A \Rightarrow^* B\) by a sequence of unit productions only.

• **Basis:** Surely \((A, A)\).

• **Induction:** If we have found \((A, B)\), and \(B \rightarrow C\) is a unit production, then add \((A, C)\).
Proof that we find exactly the right pairs

- By induction on the order in which pairs (A,B) are found, we can show $A \Rightarrow^* B$ by unit productions.
- Conversely, by induction on the number of steps in the derivation by unit productions of $A \Rightarrow^* B$, we can show that the pair (A,B) is discovered.
- Left as exercises.
Proof: Unit Production Elimination Algorithm

- **Basic idea:** there is a leftmost derivation $A \Rightarrow_{lm}^* w$ in the new grammar if and only if there is such a derivation in the old.
- A sequence of unit productions and a non-unit production is **collapsed** into a **single production** of the new grammar.
Recap: Useless Symbols

- A symbol is **useful** if it appears in some derivation of some terminal string from the start symbol.
- Otherwise it is **useless**. Eliminate all useless symbols by:
  1. Eliminating symbols that derive no terminal string.
  2. Eliminating unreachable symbols.
Cleaning Up a Grammar

Theorem

If \( L \) is a CFL, then there is a CFG for \( L - \{\varepsilon\} \) that has:

- No useless symbols.
- No \( \varepsilon \)-productions.
- No unit productions.

i.e., every right side is either a single terminal or has length \( \geq 2 \).
Proof: Start with a CFG for $L$.
Perform the following steps in order:

1. Eliminate $\varepsilon$-productions.
2. Eliminate unit productions.
3. Eliminate variables that derive no terminal string.
4. Eliminate variables not reachable from the start symbol.

Note: (1) can create unit productions or useless variables, so it must come first.
Chomsky Normal Form

**Definition**

A CFG is said to be in Chomsky Normal Form if every production is of one of these two forms:

- \( A \rightarrow BC \) (right side is two variables).
- \( A \rightarrow a \) (right side is a single terminal).

**Theorem**

If \( L \) is a CFL, then \( L - \{ \varepsilon \} \) has a CFG in CNF.
Proof of CNF Theorem

- **Step 1:** Clean the grammar, so every production right side is either a single terminal or of length at least 2.
- **Step 2:** For each right side $\neq$ a single terminal, make the right side all variables.
  1. For each terminal $a$ create a new variable $A_a$ and production $A_a \rightarrow a$.
  2. Replace $a$ by $A_a$ in right sides of length $> 2$. 
Example: Step 2

• Consider production $A \rightarrow BcDe$.
• We need variables $A_c$ and $A_e$ with productions $A_c \rightarrow c$ and $A_e \rightarrow e$.
  • **Note:** you create at most one variable for each terminal, and use it everywhere it is needed.
• Replace $A \rightarrow BcDe$ by $A \rightarrow BA_cDA_e$.
• **Step 3:** Break right sides longer than 2 into a chain of productions with right sides of two variables.

• A → BCDE is replaced by A → BF, F → CG and G → DE.
  - **Note:** F and G must be used nowhere else.

• In the new grammar, A ⇒ BF ⇒ BCG ⇒ BCDE.

• **More importantly:** Once we choose to replace A by BF, we must continue to BCG and BCDE.
  - Because F and G have only one production.
Proof of CNF Theorem (Formally)

- We must prove that Steps 2 and 3 produce new grammars whose languages are the same as the previous grammar.
- Proofs are of a familiar type and involve inductions on the lengths of derivations.
  - Left as exercises.
Pushdown Automata

- A **PDA** is an automaton equivalent to the CFG in language-defining power.
- Only the nonterministic PDA’s define all possible CFL’s.
- But the deterministic version models **parsers**.
  - **Most** programming languages have deterministic PDA’s.
Think of an $\varepsilon$-NFA with the additional power that it can manipulate a stack.

Its moves are determined by:

1. The current state (of its NFA).
2. The current input symbol (or $\varepsilon$), and
3. The current symbol on top of its stack.
• Being \textit{nondeterministic}, the PDA can have a \textit{choice} of next moves.

• In each choice, the PDA can:
  1. Change state, and also
  2. Replace the \textit{top symbol} on the stack by a sequence of \textit{zero or more} symbols.
     - \textit{Zero symbols} = \textit{pop}.
     - \textit{Many symbols} = sequence of \textit{pushes}. 
A PDA is described by:

1. A finite set of states \((Q, \text{typically})\).
2. An input alphabet \((\Sigma, \text{typically})\).
3. A stack alphabet \((\Gamma, \text{typically})\).
4. A transition function \((\delta, \text{typically})\).
5. A start state \((q_0, \text{in } Q, \text{typically})\).
6. A start symbol \((Z_0, \text{in } \Gamma, \text{typically})\).
7. A set of final states \((F \subseteq Q, \text{typically})\).
Conventions

- \(a, b, \ldots\) are input symbols.
  - But sometimes we allow \(\epsilon\) as a possible value.
- \(\ldots, X, Y, Z\) are stack symbols.
- \(\ldots, w, x, y, z\) are strings of input symbols.
- \(\alpha, \beta, \ldots\) are strings of stack symbols.
The Transition Function

- Takes three arguments:
  1. A state in $Q$.
  2. An input which is either a symbol in $\Sigma$ or $\varepsilon$.
  3. A stack symbol in $\Gamma$.

- $\delta(q,a,Z)$ is a set of zero or more actions of the form $(p,\alpha)$.
  - $p$ is a state, $\alpha$ is a string of stack symbols.
If $\delta(q,a,Z)$ contains $(p,\alpha)$ among its actions, then one thing the PDA can do in state $q$, with $a$ at the front of the input, and $Z$ on top of the stack is:

1. Change the state to $p$.
2. Remove $a$ from the front of the input (but $a$ may be $\varepsilon$).
3. Replace $Z$ on the top of the stack by $\alpha$. 
Design a PDA to accept \( \{ 0^n1^n \mid n \geq 1 \} \).

The states:
- \( q = \) start state. We are in state \( q \) if we have seen only 0’s so far.
- \( p = \) we’ve seen at least one 1 and may now proceed only if the inputs are 1’s.
- \( f = \) final state; accept.
Example: PDA

- The stack symbols:
  - $Z_0 =$ start symbol. Also marks the bottom of the stack, so we know we have counted the same number of 1’s as 0’s.
  - $X =$ marker, used to count the number of 0’s seen on the input.
• The transitions:
  
  - $\delta(q,0,Z_0) = \{(q,XZ_0)\}$. 
  - $\delta(q,0,X) = \{(q,XX)\}$. These two rules cause one $X$ to be pushed onto the stack for each 0 read from the input. 
  - $\delta(q,1,X) = \{(p,\varepsilon)\}$. When we see a 1, go to state $p$ and pop one $X$. 
  - $\delta(p,1,X) = \{(p,\varepsilon)\}$. Pop one $X$ per 1. 
  - $\delta(p,\varepsilon,Z_0) = \{(f,Z_0)\}$. Accept at bottom.
Actions of the Example PDA
Actions of the Example PDA

0 0 1 1 1

0

0

X

Z_0
Actions of the Example PDA
Actions of the Example PDA

\begin{itemize}
\item $q$
\item $X$
\item $1$ $1$ $1$
\item $X$
\item $X$
\item $X$
\item $Z_0$
\end{itemize}
Actions of the Example PDA
Actions of the Example PDA
Actions of the Example PDA

Z_0 \xrightarrow{f} f