

Test Review

Know the following:

- How to build a context-free grammar and PDA
- How to describe a PDA in English
- How to formally define a PDA
- How to describe a PDA given a set of transition functions
- How to build and understand parse trees
- How to derivate strings from context-free grammars
- When a context-free grammar is ambiguous and how to fix it.
- Pumping Lemma
- The closure properties of context-free languages
- How to convert a context-free grammar to Chomsky Normal Form.

Today's topics

- Revisit on how to construct a grammar in Chomsky Normal Form given a context-free grammar.
- Section 3.8 Deterministic PDA's

Chomsky Normal Form ($A \rightarrow BC$ or $A \rightarrow a$) where uppercase letters are non-terminals and lower case letters are terminals.

Suppose we have a context-free grammar with the following rules (The ' \mid ' represents another rule for the same non-terminal.)

$S \rightarrow AB$
 $A \rightarrow aAb \mid B \mid e$
 $B \rightarrow BA \mid e$

Our goal is to convert the above grammar G into the grammar representing Chomsky Normal form, G' , such that $L(G') = L(G) - \{e\}$.

Step 1 (Eliminate the e -link rules):

- Find E where E is a set of erasable non-terminals such that $E = \{ A \mid A^* \rightarrow e \}$, which in this case is $\{A, B, S\}$.

- Add the following rules to replace for the e -link rules that are about to be removed. These rules are determined by replacing the non-terminal in each erasable term and adding the new rules generated. Add the rules:
 - $S \rightarrow A \mid B$
 - $A \rightarrow ab$
 - $B \rightarrow A$

Note that $B \rightarrow B$ doesn't need to be added to itself.

- Remove the e -link rules.
 - $S \rightarrow e$ (by inheritance)
 - $A \rightarrow e$
 - $B \rightarrow e$
- The result from **Step 1** is:
 - $S \rightarrow AB \mid A \mid B$
 - $A \rightarrow aAb \mid B \mid ab$
 - $B \rightarrow BA \mid A$

Step 2 (Eliminate all singleton rules (*non-terminals by themselves*)):

- Find rules such that $A \rightarrow B$, where A and B are non-terminals.
- Remove all singletons and replace with non-singleton rules that exist in the non-terminal that is the singleton you are removing. So add the new rules:
 - $S \rightarrow aAb \mid ab \mid BA$ (*These are the non-singleton rules that exist inside of the singleton rules being removed from the result of non-terminal S generated from **Step 1.***)
 - $A \rightarrow BA$
 - $B \rightarrow aAb \mid ab$
- Remove all of the singletons:
 - $S \rightarrow A \mid B$
 - $A \rightarrow B$
 - $B \rightarrow A$

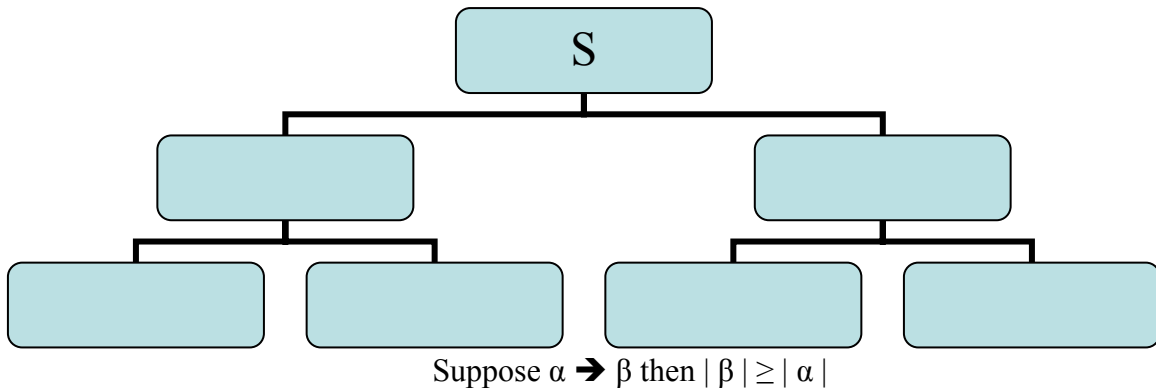
Step 3 (Convert to Chomsky Normal Form). This is a non-terminal converting into either two more non-terminals or one terminal. These are the only acceptable forms.

- $S \rightarrow AB \mid X_a X_{Ab} \mid BA$
- $A \rightarrow X_a X_{Ab} \mid X_a X_b \mid BA$
- $B \rightarrow BA \mid X_a X_{Ab} \mid X_a X_b$
- $X_a \rightarrow a$
- $X_b \rightarrow b$
- $X_{Ab} \rightarrow AX_b$

These rules are now all in Chomsky Normal Form.

Applications of Chomsky Normal Form

With Chomsky Normal Form, a grammar has binary parse trees in which every internal node on every path except the last one has exactly two children.



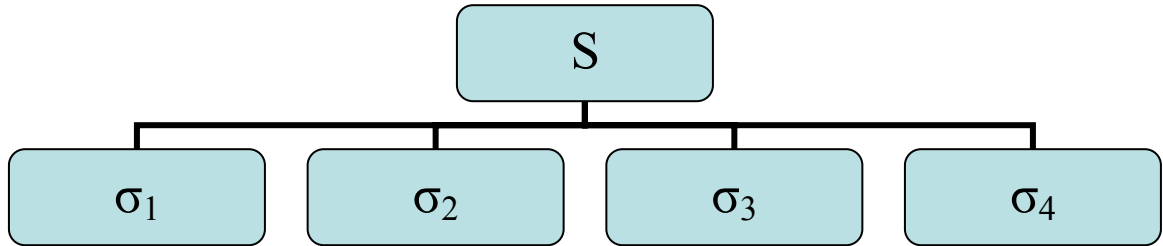
We can use this property to determine the following:

1. **Membership problem**

Given a context-free grammar G and string x . Is $x \in L(G)$?

We can convert G into Chomsky Normal Form G' . Then we want to know whether $x \in L(G')$ if $x \neq \epsilon$.

Algorithm: Search tree. Each internal node is a sentential form:
 $S \rightarrow \sigma_1 \mid \sigma_2 \mid \sigma_3 \mid \sigma_4$



As soon as x is generated, stop and output “yes”.

If a sentential form in the search tree is larger than $|x|$, then we cut off the entire sub-tree with this form as the root.

Since there are only a finite many sentential forms of length $\leq |x|$ and there are singleton rules, this process will stop. If x is generated, output “yes”, otherwise the output is no.

2. The emptiness problem

Given a context-free grammar G , is $L(G) = \emptyset$?

Algorithm:

First convert the grammar to Chomsky Normal Form G' .

If $L(G') \neq \emptyset$, then there must be a positive integer n (obtained from the lemma) s.t. $L(G')$ contains a string of length $< n$.

We can use a for loop to loop through all possible strings x of length $< n$ and check whether $x \in L(G)$ or not.

3. The finiteness problem

Given a context-free grammar G , is $L(G)$ finite?

Algorithm:

If $L(G)$ is infinite, then there must be a positive integer n such that $L(G)$ contains a string w with $n \leq |w| < 2n$.

---- END OF LECTURE ----