Chomsky Normal Form

Chomsky Normal Form

- When working with grammars, it is often convenient to have them in a simple, standard form
  - Helps for identifying if a specific string can be generated by the grammar
  - Helps for proofs
    - Context-free grammar version of the pumping lemma
    - Decidability proofs for context-free grammars
    - Complexity proofs for context-free grammars
    - (we'll get to all of these later in the course)

Chomsky Normal Form

- A context-free grammar is in Chomsky Normal Form (CNF) if every production (rule) is of the form:
  - $A \rightarrow BC$
  - $A \rightarrow a$
  - $S \rightarrow \epsilon$

  - Where A, B, and C are variables (B and C can not be the starting variable)
  - And where a is a terminal.
  - And where S is the starting variable (to allow empty string)

Chomsky Normal Form

- Theorem: Any context-free language is generated by a context-free grammar in CNF
- Equivalently:
  - Any context-free grammar can be converted to CNF
- We'll do a proof by construction: defining rules for converting an arbitrary context-free grammar into a context-free grammar in CNF

CNF Conversion Steps

- Step #1: add a new start variable and a rule
  - $S_0 \rightarrow S$ where S is the original starting variable
  - What does this do?
    - It eliminates any possibility of the starting variable occurring on the right hand side of any rule

CNF Conversion Steps

- Step #2: remove all ϵ rules
  - Remove $A \rightarrow \epsilon$ where A is any variable (other than the new starting variable)
  - This may take several sub-steps.
    - Each time we remove such a rule $A \rightarrow \epsilon$, we look for all occurrences of variable A on the right hand side in other rules, and add new rules in which A is replaced with ϵ
    - If A appears by itself on the right hand side of any rule, this creates a new example of a $B \rightarrow \epsilon$ rule that also needs to be fixed
  - When we're done – we've removed ϵ from all rules (except $S_0 \rightarrow \epsilon$ if it got created)

CNF Conversion Steps

- Step #3: remove all unit rules
  - $A \rightarrow B$
  - This also may take several sub-steps.
    - We can get rid of empty unit rules $A \rightarrow A$
    - Each time we remove such a rule $A \rightarrow B$, we add new rules $A \rightarrow u$ for each existing $B \rightarrow u$ rule (unless this adds back a rule we've already eliminated)
  - When we're done with this step, all rules will involve either a single terminal or multiple symbols (variables and terminals combined)

This step may cause a variable to become unreachable. It can be eliminated.

CNF Conversion Steps

- Step #4: convert rules into proper format
  - Rules with too many symbols can be broken down by introducing intermediate variables, e.g.:
    - A → BCD becomes
      - A → BE
      - E → CD
    - A → aB becomes
      - A → CB
      - C → a
  - When we're done with this step, we're in CNF!
  - (see Sipser pages 108-111 for more details)

CNF – (Sipser Example 2.10)

- Starting grammar G
  - $S \rightarrow ASA \mid aB$
  - $A \rightarrow B \mid S$
  - $B \rightarrow b \mid \epsilon$

- (We'll do this on the board)

CNF – Example (Sipser 2.14)

- Starting grammar G
  - $A \rightarrow BAB \mid B \mid \epsilon$
  - $B \rightarrow 00 \mid \epsilon$

- Try this one on your own
  - (answer on the following slide)

CNF – Example (Sipser 2.14)

- Starting grammar G
  - $A \rightarrow BAB \mid B \mid \epsilon$
  - $B \rightarrow 00 \mid \epsilon$

- $S_0 \rightarrow AB \mid CC \mid BA \mid BD \mid BB \mid \epsilon$
- $A \rightarrow AB \mid CC \mid BA \mid BD \mid BB$
- $B \rightarrow CC$
- $C \rightarrow 0$
- $D \rightarrow AB$

Using CNF

- Given a CFG, G, how can we determine if a particular string w is generated by G?
  - For an arbitrary CFG, G, it is not obvious how to know when all possibilities have been tried
  - What about G' in CNF?
    - String w has length n
    - What must be true about any string of length n generated by G' ?
      - It uses n-1 applications of a rule of the type $A \rightarrow BC$
      - It uses n applications of a rule of the type $A \rightarrow a$
        - All derivations of strings of length $n>0$ involve exactly $2n-1$ derivation steps – we can exhaustively search them all