CS 344 - Context Free Languages

Announcements

- HW due Friday

7329 \leq \text{decimal}; \text{ no left } 0

0x7341 \leq \text{ hexadecimal}

0 \text{ octal}
A demo or two

flex

flex filename lex

gcc lex.yy.c -lfl
Context-Free Languages

Recall that for any context-free languages, there are an infinite number of grammars that can produce it.

We wish to somehow give a definition of a "good" set of productions.

Goal: Parsing (well) - given a language, detect if a string is in that language.
Ex: **Bad example**

\[ S_0 \rightarrow S | x | z | A \]

- **B** is **useless**
- **Y** is **unreachable**

\[ S \rightarrow A \]
\[ A \rightarrow B \]
\[ C \rightarrow A_a \]
\[ X \rightarrow C \]

\[ Y \rightarrow aY | a \]
\[ z \rightarrow \varepsilon \]
Chomsky Normal Form (CNF)

Each rule in the grammar is either:

* A → BC

  where neither B or C is the start variable, or both are nonterminals

* A → a

  where a is a terminal

* S → ε

  where S is the start symbol
Thm: All grammars can be converted to CNF.

Procedure:

1. Create a new start symbol \( S_0 \), a send \( S_0 \rightarrow S \)

Eliminate useless rules

(Just delete ones that can't be reached)
2) Remove nullable variables. $A \rightarrow \varepsilon$

How?
Remove all $\varepsilon$ productions.
Then fix.

$A \rightarrow CBC | CC$

$\vdash$ $B \rightarrow \varepsilon$

$A \rightarrow \varepsilon \Leftarrow$ to remove, just add another rule to anything with $A$ on right hand side
③ Remove unit rules:

\[ S \rightarrow A \]

How? Must have:

\[ X \rightarrow Z_i, \quad Z_i \rightarrow Z_j, \quad \ldots, \quad Z_k \rightarrow Y \]

(Since we removed \( \varepsilon \)-transitions in ②)

Find all these pairs \( X \Rightarrow Y \)

\[ Y \rightarrow a \quad b \quad c \quad d \]

\[ Y \rightarrow aB \]
For each unit pair \((A, B)\) and rule \(B \rightarrow w\), \(B \rightarrow YZ\), .

add \(A \rightarrow w\) to a new grammar.

(Note that \((A, A)\) is a unit pair, so all rules \(A \rightarrow w\) will stick around.)
4. Get rid of "long" right-hand sides.

4a: Create $V_c \rightarrow c$ for every character.

Replace $c$ with $V_c$ everywhere.

Now either

$A \rightarrow CDEF$

or

$V_c \rightarrow c$. 
46: $A \rightarrow B_1 B_2 B_3 \ldots B_k$

How to replace with only 2 nonterminals on the right?
Ex:

\[ S \rightarrow ASA \mid aB \]
\[ A \rightarrow B \mid S \]
\[ B \rightarrow b \mid \epsilon \]

1. Removing \( \epsilon \):

\[ S \rightarrow ASA \mid aB \mid a \mid AS \mid SA \]
\[ A \rightarrow S \mid B \]

2. Removable pairs: \( (A, B), (A, S) \)
Ex (cont.):
Now—why do we care??

Parsing: building those parse trees we saw

In general, there are an exponential number of parse trees for a given input.

So how to check quickly?

Even in CNF, there might be 2^n possible parse trees.
Cocke–Younger–Kasami (CYK) algorithm

Uses a table & dynamic programming
to give a parse tree in \(O(n^3)\) time.

Grammar must be in CNF!