Announcements

1. HW due Saturday
- Bison
-Parser tools
- Context free languages

Last time - Ch 2. Sec 3
Examples: Palindromes

Recursive defn: First & last letters
match & interior is a palindrome.

s <--- aSa
1st 2nd 3rd
\[ \text{Call} \]

\[ \text{if} \]

\[ - 1513 \rightarrow 5 \]

Strings of even or all zeros.

Devise a CF for all binary
An alternative for that last one:
Context-Free Languages

Recall that for any context free language 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.
A parse tree packet:

- If it is valid, compile.
- Need to see if it is a complete goal.

Remember, goal is to compile a program.

Why?
\[ x \rightarrow y \rightarrow z \]

- Halft-pair
- Non-terminals
- Unreachable
- Useless states

Why is it bad?

So - a bad example: So \( s \leftarrow s \times z \)
Symbol $S$ is the start

where $a$ is a nonterminal

where $a$ is a terminal

Start variable $A$ or $C$ is the

where variable $A$ or $C$ is either:

Chomsky Normal Form (CNF)
But: Clearly, not always in CNF.

Luckily:

Thm: All grammars can be converted to CNF.
1. Eliminate useless rules
2. Create a new shift symbol
3. Send S 5 new shift symbol, So
4. Can't be reached
D → BACB | D → BACB | A → CB | C

\[
\begin{align*}
&\text{B → } C \\
&\text{A → CB} \\
&\text{A → } \overline{CB} \\
\end{align*}
\]

Then, if, then all \& prove proofs.

Remove nullable variables.

\( \) Remove nullable variables.
Then:

\[
\begin{align*}
S &\rightarrow Axy \\
S &\rightarrow Xy \\
S &\rightarrow B/c \\
A &\rightarrow B/c \\
A &\rightarrow Xy \\
X &\rightarrow Z_1 \quad \vdots \\
X &\rightarrow Z_k
\end{align*}
\]

Since we removed ε-transitions in (2):

\[X \rightarrow Z_1 \quad \vdots \quad X \rightarrow Z_k\]

Then, it must have:

\[S \rightarrow A \rightarrow S\]

(3) Remove unit rules:
Note so that (A, A) is a unit pair.

Add A → w to a new grammar.

For each unit pair (A, B), add w → B w.

(Still 3)
or $V_0 \rightarrow c$.

$A \rightarrow \text{ Concept}$

Now erases.

Replace a with $V_0$ everywhere.

Character:

4a: Create $V_0 \rightarrow c$ for every side of "long" right hand

$\text{Get rid of "long" right hand}$

○
\[
\begin{align*}
& \frac{c}{x^2} + \frac{1}{B} = \frac{z}{y} \\
& B = x' \\
& A = B \times \cdot \\
& A \rightarrow B^2 \times^2 \\
& x^2 \rightarrow B_{-1} \\
& x^2 \rightarrow B, x^3 \\
& x' \rightarrow B^2, x^2 \\
& A \rightarrow B, x \\
& \text{right?}
\end{align*}
\]
Ex.

\[ S \triangle ASA \sim \triangle B \]

\[ A \rightarrow B \]

\[ \triangle 3 \]
Even in CFE might be do possible

So how to check quickly?

For a given input:

In general, there are no exponential

Parsing: building those parse trees

Now—why do we care?
Grammar must be in CNF!

Chop first parse tree in pre-order.

Uses a table or dynamic programming.

Cook - Younger - Kasami (CYK) algorithm.
We'll build a table from the bottom up. But first, we'll parse.

- Start by looking at how they can go cup...
- Substrings we possible...
- We'll look at all possible...

Given a word w = w_1w_2...w_n...

---

CKK Algorithm
Running time:
Say we have $n$ rules.
Converting to CNF:
$O(n^2)$

Running CYK: $n^3$ time, $n^2$ space
Two big (or useful) classes have linear
grammars could. Having detest algorithms
was put into. Advertising, a lot of work.
After it was developed, a lot of work.

For large programming languages,
CIL is still pretty dumb, especially
other parsing algorithms