UMB CS 622

Non-CFLs

Wednesday, March 27, 2024
Announcements

• HW 6
  • Due Monday 4/1 12pm noon
Application of this class: Compilers

A program string (chars) (e.g., \( \text{a} : = (5 + 3) ; \ldots \))

DFAs (recognizing regular languages) in here!

Program "words" (e.g., ID(a) ASSIGN LPAREN NUM(5) PLUS NUM(3) RPAREN SEMI ...)

Last Time
Application of this class: Compilers

A program string (chars) (e.g., \( a = (5 + 3) \); ...)

DFAs (recognizing regular languages) in here!

Program “words” (e.g., ID(a) ASSIGN LPAREN NUM(5) PLUS NUM(3) RPAREN SEMI ...)

DPDAs (recognizing DCFLs) in here!

Syntax tree (AST), i.e., a parse tree!

AssignStm
  a
   OpExp
     NumExp
       5
     Plus
     NumExp
       3
Subclasses of CFLs

2 parser design decisions:
1) Parse from left, or from right

2) choose “look ahead” amount

DCFLs

Programming language parsers / compilers are ideally in here
To learn more, take a Compilers Class!

This phase needs computation that goes beyond CFLs
Flashback: Pumping Lemma for Regular Langs

- **Pumping Lemma** describes how strings repeat

- A non-regular language: 
  \[ \{0^n1^n \mid n \geq 0\} \]

  Kleene star can’t express this pattern: 2\textsuperscript{nd} part depends on (length of) 1\textsuperscript{st} part

- Q: How do CFLs repeat?
Repetition and Dependency in CFLs

Parts before/after repetition point linked (not independent)

\[ A \rightarrow 0A1 \]
\[ A \rightarrow B \]
\[ B \rightarrow \# \]

\[ \{0^n#1^n \mid n \geq 0\} \]

Repetition

A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000#111
How Do Strings in CFLs Repeat?

- Strings in CFLs repeat subtrees in the parse tree.

One repeated subtree means that it can be repeated any number of times.

Linked parts repeat together.

5 substrings.
Pumping Lemma for CFLS

Pumping lemma for context-free languages  If $A$ is a context-free language, then there is a number $p$ (the pumping length) where, if $s$ is any string in $A$ of length at least $p$, then $s$ may be divided into five pieces $s = uvxyz$ satisfying the conditions

1. for each $i \geq 0$, $uv^i xy^i z \in A$,
2. $|vy| > 0$, and
3. $|vxy| \leq p$.

Pumping lemma  If $A$ is a regular language, then there is a number $p$ (the pumping length) where if $s$ is any string in $A$ of length at least $p$, then $s$ may be divided into three pieces, $s = xyz$, satisfying the conditions

1. for each $i \geq 0$, $xy^i z \in A$,
2. $|y| > 0$, and
3. $|xy| \leq p$. 
A Non CFL example

\[ \text{language } B = \{a^n b^n c^n | n \geq 0\} \text{ is not context free} \]

Intuition

• Strings in CFLs can have **two parts** that are “pumped” together
• Language \( B \) requires **three parts** to be “pumped” together
• So it’s not a CFL!

Proof?
Want to prove: $a^n b^n c^n$ is not a CFL

Proof (by contradiction):

• **Assume:** $a^n b^n c^n$ is a CFL
  • So it must satisfy the pumping lemma for CFLs
  • I.e., all strings $\geq$ length $p$ are pumpable
• **Counterexample** = $a^p b^p c^p$

Choose $x$, split so $y$ contains:
all $0$s

Pumping $y$: produces a string with more $0$s than $1$s
Which is not in the language $0^p 1^p$
This means that $0^p 1^p$ does not satisfy the pumping lemma
Which means that that $0^p 1^p$ is a not regular language
This is a contradiction of the assumption!

**Pumping lemma for context-free languages**

If $A$ is a context-free language, then there is a number $p$ (the pumping length) where, if $s$ is any string in $A$ of length at least $p$, then $s$ may be divided into five pieces $s = uvxyz$ satisfying the conditions

1. For each $i \geq 0$, $uv^i x y^i z \in A$
2. $|vxy| > 0$, and
3. $|vxy| \leq p$.

**Reminder:** CFL Pumping lemma says:
all strings $a^n b^n c^n \geq$ length $p$ are splittable into $uvxyz$ where $v$ and $y$ are pumppable

Contradiction if:
- $A$ string in the language
- $\geq$ length $p$
- Is not splittable into $uvxyz$ where $v$ and $y$ are pumpable
Want to prove: $a^n b^n c^n$ is not a CFL

Possible Splits

Proof (by contradiction):

- **Assume:** $a^n b^n c^n$ is a CFL
  - So it must satisfy the pumping lemma for CFLs
    - i.e., all strings $\geq$ length $p$ are pumpable
  - **Counterexample:** $a^p b^p c^p$

- **Possible Splits** (using condition # 3: $|vxy| \leq p$
  - $vxy$ is all $a$s
  - $vxy$ is all $b$s
  - $vxy$ is all $c$s
  - $vxy$ has $a$s and $b$s
  - $vxy$ has $b$s and $c$s
  - ($vxy$ cannot have $a$s, $b$s, and $c$s)

So $a^n b^n c^n$ is not a CFL

(justification: contrapositive of CFL pumping lemma)

---

Pumping lemma for context-free languages

If $A$ is a context-free language, then there is a number $p$ (the pumping length) where, if $s$ is any string in $A$ of length at least $p$, then $s$ may be divided into five pieces $s = uvxyz$ satisfying the conditions

1. for each $i \geq 0$, $uv^i xy^i z \in A$,
2. $|vxy| > 0$, and
3. $|vxy| \leq p$.

---

$uvxyz$ cannot be split into $uvxyz$ where $v$ and $y$ are pumpable

$vxy$ ???
Another Non-CFL \( D = \{ww \mid w \in \{0,1\}^*\} \)

Be careful when choosing counterexample \( s \): \( 0^p10^p1 \)

This \( s \) can be pumped according to CFL pumping lemma:

• CFL Pumping Lemma conditions:
  1. for each \( i \geq 0 \), \( uv^i xy^i z \in A \),
  2. \( |vy| > 0 \), and
  3. \( |vxy| \leq p \).

So this attempt to prove that the language is not a CFL failed. (It doesn’t prove that the language is a CFL!)
Another Non-CFL \( D = \{ww \mid w \in \{0,1\}^*\} \)

- Need another counterexample string \( s \): If \( vyx \) is contained in first or second half, then any pumping will break the match.  
  \[0^p 1^p 0^p 1^p\]
  So \( vyx \) must straddle the middle
  But any pumping still breaks the match because order is wrong

- CFL Pumping Lemma conditions:
  1. for each \( i \geq 0 \), \( uv^i xy^i z \in A \),
  2. \( |vy| > 0 \), and
  3. \( |vxy| \leq p \).

Now we have proven that this language is not a CFL!
A Practical Non-CFL

- **XML**
  - ELEMENT $\rightarrow$ <TAG>CONTENT</TAG>
  - Where TAG is any string

- XML also looks like this non-CFL: $D = \{ww \mid w \in \{0,1\}^*\}$

- This means XML is not context-free!
  - **Note:** HTML is context-free because ...
  - ... there are only a finite number of tags,
  - so they can be embedded into a finite number of rules.

In practice:
- XML is parsed as a CFL, with a CFG
- Then matching tags checked in a 2nd pass with a more powerful machine ...
Next: A More Powerful Machine ...

$M_1$ accepts its input if it is in language: $B = \{ w\#w \mid w \in \{0,1\}^* \}$

$M_1 =$ “On input string $w$:

1. Zig-zag across the tape to corresponding positions on either side of the # symbol to check whether these positions contain the same symbol. If they do not, or if no # is found, reject. Cross off symbols as they are checked to keep track of which symbols correspond.

Can move to, and read/write from arbitrary memory locations!

Infinite memory (initial contents are the input string)