CS 3313
Foundations of Computing:

Examples of use of CFL Pumping Lemma

Going Back to Before the Exam

- Today we will practice using the CFL pumping lemma
- You will need this on the next homework

Statement of the CFL Pumping Lemma

For every context-free language L

There is an integer p, such that

For every string s in L of length $\geq$ p

There exists s = uvwxy such that:

1. $|vwx| \leq p$.
2. $|vx| > 0$.
3. For all $i \geq 0$, uv^iwx^iy is in L.

- You cannot fix the value of p
- vwx can fall anywhere in the string as long as it satisfies $|vwx| \leq p$

=> have to consider all cases for vwx

L_1: { a^m | *m is a prime number*}

1. Assume it is CFL and let p be the constant of the lemma
2. Pick $z= a^n$ where n is the smallest prime larger than p
3. $z= uvwxy$
   - All the substrings consist entirely of a's
   - Let $v= a^j$ and $x=a^k$
   - Remaining string *uwy* consists of $n - (j+k)$ a's.

- From lemma, we know that $1 \leq j+k \leq p$

L_1: { a^m | m is a prime number}

- From lemma, uv^iwx^iy is in L_1 for all $i \geq 0$
  - we need to pick an i so that the resulting number of a's are not prime.
- How to get a contradiction: pick a value of *i* such that we end up with a number that can be factored
- Pick *i*= *n+1, since vx consists of (j+k) a's*
  - $uv^iwx^iy = a^{n-(j+k)}\, a^{(n+1)(j+k)} = a^{(n-(j+k)+(n+1)(j+k))}$
- *So, the number of a's is* $m = (n-(j+k)+(n+1)(j+k)) = n+n(j+k) = n(1+j+k)$.
- *Since* $(j+k) \geq 1$, $(1+j+k) \geq 2$
- *Therefore* $m=n(1+j+k)$ *is not a prime*
  - *Since it has two factors, both greater than 1.*

L_2: { $w \mid w \in \{a,b,c\}^*$, and $n_a(w) = n_b(w) * n_c(w)$ }

- This language does not place restrictions on the pattern
  - $n_a(w)$= number of a’s in the string w, etc.
  - We can have a’s after b’s etc.
- Intuition: we need to keep track of number of b’s and c’s, and then multiply the two…this implies we need to store two variables ($n_b(w)$ and $n_c(w)$): likely not context free

L_2: { *w* | *w* ∈ {*a*,*b*,*c*}*, *and* $n_a(w)= n_b(w)*n_c(w)$ }

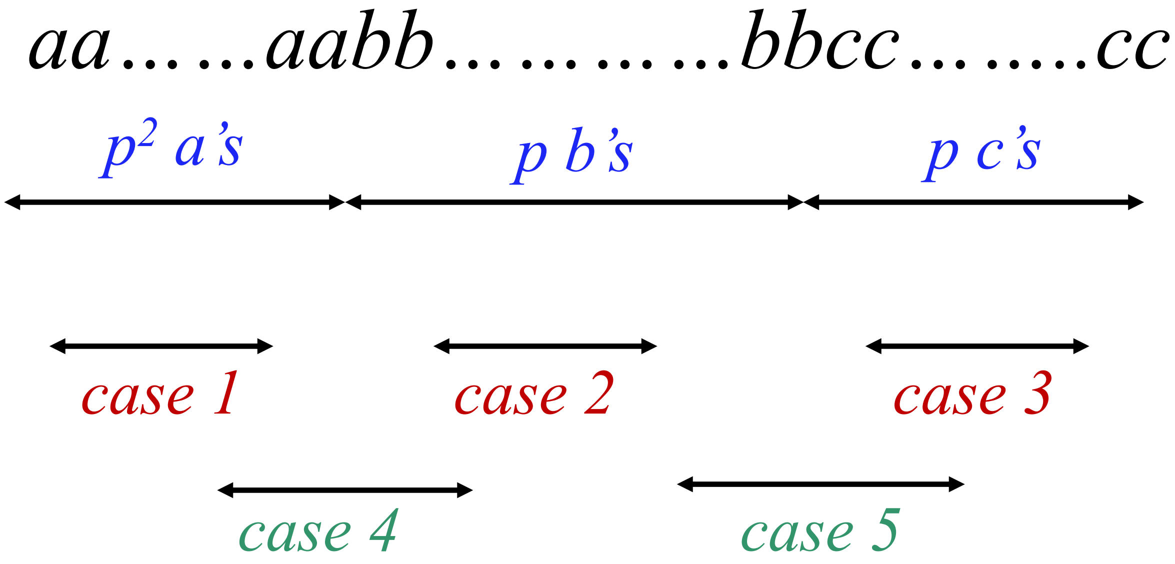
- Assume L_2 context free, let *p* be the constant of the lemma
- We need to pick values for $n_a(w)$, $n_b(w)$, $n_c(w)$ which will make it easy to prove the $n_a(w)$ in pumped string cannot be the product of $n_b(w)$ and $n_c(w)$
- Additionally, pick a pattern that makes it easier to determine the different cases of *vwx*

L_2: $\{ w \mid w \in \{a,b,c\}^*$, and $n_a(w) = n_b(w) * n_c(w)$

- Let p be the constant and pick $z = a^m b^p c^p$ where $m=p^2$
  - *why pick this as z ?*
  - We want to construct an instance of $n_b(w) * n_c(w)$ which will make it easier to contradict: if we pick perfect squares then we know that the next perfect square after p^2 *is* $(p+1)^2$ which is $(2p+1)$ more than p^2
  - Lemma states, $|vwx| \leq p$ *and* $|vx| \geq 1$
- Next: look at the possible cases for where vwx could be
  - We need to find a contradiction for each of these cases

L_2: { w | $w \in \{a,b,c\}^*$, and $n_a(w)= n_b(w) * n_c(w)$

- Let’s look at the possible cases for where vwx could be
  - We need to find a contradiction for each of these cases



$aa\ldots\ldots aabb\ldots\ldots\ldots\ldots bbcc\ldots\ldots cc$

p^2 a’s

p b’s

p c’s

case 1

case 2

case 3

case 4

case 5

Observation:
vx in cases 1,2,3 consist of one type of symbol/terminal
vx in cases 4,5 consists of two types of symbols

Cases 1, 2, and 3

L_2: { *w* | *w* ∈ {*a,b,c*}*, *and* $n_a(w) = n_b(w) * n_c(w)$

- Case 1: *vx* consists entirely of a's => $v = a^j$, $x = a^k$
- From Lemma: $(j+k) \geq 1$ *and* $(j+k) \leq p$
- *Consider* $z' = uv^2wx^2y = a^{\{p^2+(j+k)\}}b^pc^p$
    - *But,* $n_b(w) = n_c(w) = p$
    - *Since* $p^2 + (j+k) > p^2$, *we know that* $n_a(w) \neq n_b(w) * n_c(w)$
- *Therefore z' it is not in the* L_2

- Cases 2, 3: *i.e. vx consists entirely of b's or entirely of c's*
    - *Setting i=2, we get an increase in either the number of b's or c's without increasing a's. So,* $n_a(w) \neq n_b(w) * n_c(w)$

Cases 4,5

$L_2 = \{ w \mid w \in \{a,b,c\}^*, \text{ and } n_a(w) = n_b(w) * n_c(w)$

- Cases 4,5 are a bit more complicated
- if either *v* or *x* consist of two different symbols then uv^2wx^2y will have a’s after b’s etc…..but this is allowed in this language!!
- Case 4: *vx* consists of *j* a’s and *k* b’s – we don’t care about the exact pattern
- Case 5: *vx* consists of *j* b’s and *k* c’s – we don’t care about the exact pattern

Cases 4,5

$L_{2=}\{ w \mid w \in \{a,b,c\}^*, \text{ and } n_a(w) = n_b(w) * n_c(w)$

- Case 4: *vx* consists of *j* a's and *k* b's – we don't care about the exact pattern
- From conditions of the lemma, $(j+k) > 0$ *and* $(j+k) \leq p$
- Therefore, z' = uv^2wx^2y will have
  - $n_a(z') = (p^2 + j)$
  - $n_b(z') = (p + k)$
  - $n_c(z') = p$
- Question: is $(p^2 + j) = p(p+k)$?
  - If $p^2 + j = p^2 + pk$ then $j = pk$
    - If $k=0$ then $j=0$ **contradiction since** $(j+k)>0$
    - If $k>0$ then $j = pk \geq p$, *so* $(j+k) > p$ **contradiction** since $(j+k) \leq p$

- Case 5 is similar

Exercise:
$L_{3} = \{ x\, w\, w^R\, y \mid x=y,\ x,y \in \{0,1\}^*,\ w \in \{a,b\}^* \}$

- Intuition: While recognizing ww^R can be done using a stack, recognizing $x=y$ implies a stack storage is not sufficient
  - This property is like the language *ww* – see earlier proof (and in textbook) that it is not context free.

- Application of pumping lemma now requires carefully choosing the string so we can simplify the proof and focus in on what seems to be the non-context free property of $x=y$.
- Assume it is CFL and let p be the constant of the lemma

L_3: $\{\, x\, w\, w^R\, y \mid x{=}y,\ x,y \in \{0,1\}^*,\ w \in \{a,b\}^* \,\}$

- **Hint**: what is the smallest string that *w* can be ? What does a string *z* look like with this smallest “value” for *w* ?

- Next: write out this string and consider the different cases.