CS 3313
Foundations of Computing:

Examples of use of CFL Pumping Lemma

http://gw-cs3313-2021.github.io

Statement of the CFL Pumping Lemma

For every context-free language L

There is an integer n, such that

For every string z in L of length $\geq$ n

There exists z = uvwxy such that:

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

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

=> have to consider all cases for vwx

L_1: $\{ a^i \mid i \text{ is a prime number}\}$

- Intuition: We need to run some kind of algorithm that has to remember which primes have been checked with i.
- Application of pumping lemma similar to proof that this language is not regular – and we only have one case for splitting the string into uvwxy
- Assume it is CFL and let n be the constant of the lemma
- Pick $z = a^p$ where p is the smallest prime larger than n
- $z = uvwxy$
  - All the substrings consist entirely of a's
  - Let $v = a^j$ and $x = a^k$ (v consists of j *a's* and x consists of k *a's*)
  - Remaining string *uwy* consists of $p - (j+k)$ a's.
- From lemma, $1 \leq j+k \leq n$

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

- From lemma, uv^iwx^iy is in L_1 for all $i \geq 0$
  - Similar to how we proved it is not regular, we pick an i so that the resulting number of a's are not prime.
- Pick $i = p+1$
- $uv^iwx^iy = a^{p-(j+k)}\, a^{(p+1)(j+k)} = a^{(p-(j+k)+(p+1)(j+k)}$
- $m = (p-(j+k) + (p+1)(j+k) = p + p(j+k) = p\,(1+j+k)$.
- *Since* $(j+k) \geq 1$, $(1+j+k) \geq 2$
- *Therefore* $m=p(1+j+k)$ *is not a prime*
  - *Since it has two factors, both greater than 1.*

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

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

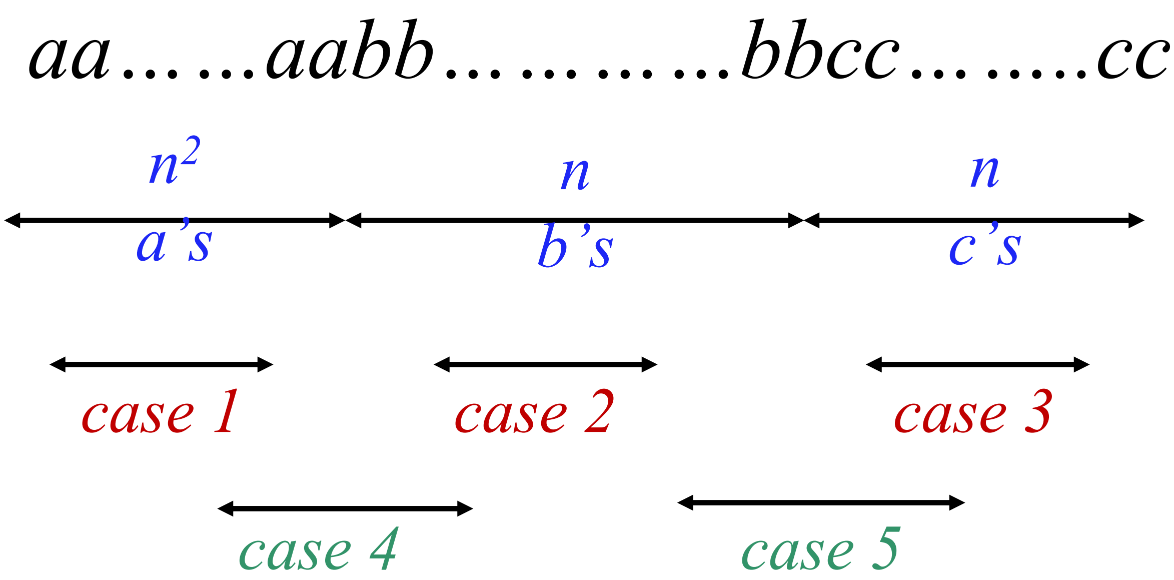
L_2: { w | w {a,b,c}*, and $n_a(w) = n_b(w) * n_c(w)$

- Let n be the constant and pick $z = a^m b^n c^n$ where $m = n^2$
  - *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 n^2 is $(n+1)^2$ which is $(2n+1)$ more than n^2
  - Lemma states, $|vwx| \leq n$ 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

aa……aabb…………bbcc……..cc

L_2: { w | w {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



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

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

- Cases 1,2,3 are similar…Let's show how to derive contradiction for one of these
  - The other two are similar
- How about cases 4,5 ?
- From the definition of the language L_2 can we have a's after b's etc. ?
  - So what happens if v or x contains two types of symbols (ex: a's and b's) and we pump the string twice ? Can we get a contradiction just because a's occur after b's ?

- To complete the proof: for each case, find value of *i,* such that

  $z' = uv^iwx^iy$ cannot be in L_2

Answer: Setting up Case 1
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* $(j+k) \leq n$
- *Consider* $z' = uv^2wx^2y = a^{n2 +(j+k)} b^n c^n$
  - *How do you get a contradiction ?*
- *Therefore z' it is not in the language*
- For Cases 2,3*: ?*

Answer: Cases 1,2,3
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* $(j+k) \leq n$
- *Consider* $z' = uv^2wx^2y = a^{n2+(j+k)} b^n c^n$
  - *Since* $(j+k) \geq 1,\ n^2+(j+k) > n^2$ *therefore* $n_a(z') <> n_b(z')*n_b(z')$
- *Therefore z' it is not in the language*
- For Cases 2,3: set $i=2$ and we get $n_a(z')=n^2$ *and*

  $n_b(z')*n_b(z')=n(n+j+k)$
  - *Since* $(j+k)>0,\ n(n+j+k) = n^2+ n(j+k) > n^2$
  - *i.e.,* $n_a(z') <> n_b(z')*n_b(z')$

Answer: Setting up Cases 4,5
$L_2 = \{ w \mid w \{a,b,c\}^*, \text{ and } n_a(w) = n_b(w) * n_c(w)$

- Cases 4,5 are a bit more complicated than in earlier examples such as $a^n b^n c^n$ or $a^n b^m c^n d^m$..
- 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!!
  - We take a more general approach now….
  - Note that these cases can be simplified if use closure properties before applying the pumping lemma
- 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
- From conditions of the lemma, $(j+k) > 0$ *and* $(j+k) \leq n$

- Consider Case 4 – Case 5 will be similar.
  - Pick i=2, and consider the string z' = uv^2wx^2y

Answer: Cases 4,5
$L_{3=}$ { w | w {a,b,c}*, 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* $(j+k) \leq n$
- Therefore, z' = uv^2wx^2y will have
  - $n_a(z') = (n^2 + j)$ *(number of a's)*
  - $n_b(z') = (n + k)$
  - $n_c(z') = n$
- Question: is $(n^2 + j) = n(n+k)$?
  - If $n^2 + j = n^2 + nk$ then $j = nk$
    - If $k=0$ then $j=0$ **contradiction since** $(j+k)>0$
    - If $k>0$ then $j = nk \geq n$ and thus $(j+k) > n$ **contradiction** since $(j+k) \leq n$

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 book for proof 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 n be the constant of the lemma

L_4: $\{ x\ w\ w^R\ y \mid x=y,\ x,y \in \{0,1\}^*,\ w \in \{a,b\}^* \}$

- **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.

Answer:

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

- **Cute trick**: since $w \in \{a,b\}^*$, we can pick $w=\lambda$ (empty string) and thus pick $z= 0^n 1^n 0^n 1^n$!!!!!
- To prove that there is an i, such that uv^iwx^iy is not in L_3 for all cases of vwx, we can use the proof that shows $L=\{ww\}$ is not context free.