CS 361 Meeting 17 — 4/8/20

Announcements

1. Homework assignment 6 is due today (if your group meets with me tomorrow) or tomorrow (if your meeting is on Friday).

Ambiguity

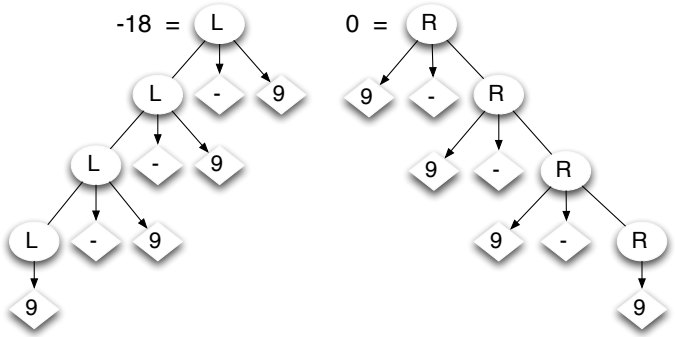
(Click for video)

1. The grammatical structure that a grammar induces on a sentential form can play a role in the way we associate meaning with the string.

   - Consider the string $9 - 9 - 9 - 9$ (which is intended to be interpreted as a very simple arithmetic expression involving three subtractions).
   - This string clearly belongs to the language of both of the following grammars:

     $L \to L - 9 \mid 9$ $\qquad\qquad$ $R \to 9 - R \mid 9$

     but the parse trees induced on $9-9-9-9$ by these two grammars suggest different interpretations of the string as an arithmetic expression:

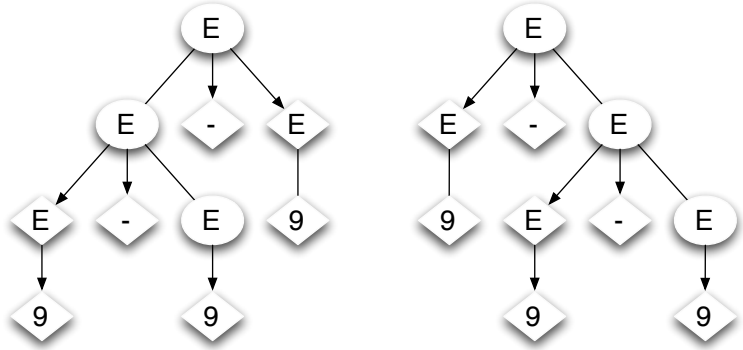


   - The tree on the left suggests the expression should be interpreted with left associativity for the subtraction operations so that the result is equal to $(((9 - 9) - 9) - 9) = -18$. The tree on the left suggests right associativity so that the result would be $(9 - (9 - (9 - 9))) = 0$.

2. In the example above, as long as we know which of the two grammars should be used we would know how to interpret input strings. There are, however, grammars for which a single string may have both multiple derivations and multiple parse trees.

   - Consider the following grammar for the sort of simple subtraction languages we have been discussing:

     $E \to E - E \mid 9$
   - Relative to this grammar, the string $9 - 9 - 9$ has two parse trees with fundamentally different structures:



   - The tree on the left suggests left-association so that $9-9-9 = (9-9) - 9 = -9$ while the tree on the right suggest right-association so that $9 - 9 - 9 = 9 - (9 - 9) = 9$.

3. If for any string w, a context-free grammar induces two or more parse trees with distinct structures, we say the grammar is *ambiguous*. Otherwise we say the grammar is *unambiguous*.

The Pumping Lemma for Context-Free Languages

(Click for video)

1. Just as we used the Pumping Lemma to prove that certain languages were not regular, there is a pumping lemma for context-free languages that provides a way to show that certain languages are not context-free.

Click here to view the slides for this class

Lemma: If C is a context-free language, there is a number p known as the pumping length such that if $w \in C$ and $|w| \geq p$, then we can partition w into five substrings $w = uvxyz$ such that:

- $|vxy| \leq p$
- $|vy| > 0$
- $\forall i \geq 0, uv^ixy^iz \in C$

2. Since a pushdown automaton only has finite states, the same argument that states must be repeated on long inputs that applies to finite automata applies to PDAs, but...

3. If a PDA enters the same state with different elements in its stack, it isn't necessarily possible to repeat the process indefinitely by just duplicating the string that brought the PDA back to a previously visited state. The "effective state" of a PDA includes both its current state and its stack contents. There are infinitely many possible stack contents, so a machine may not return to the same effective state no matter how long an input is processed.

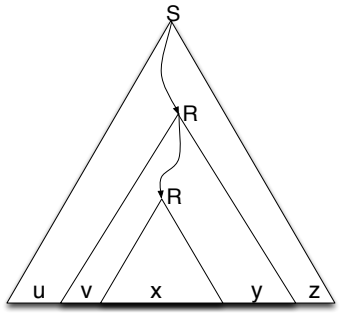
4. As a result, the Pumping Lemma for CFLs is not derived from an argument based on PDAs.

5. Instead, the proof of the Pumping Lemma for CFLs follows from two properties of parse trees:

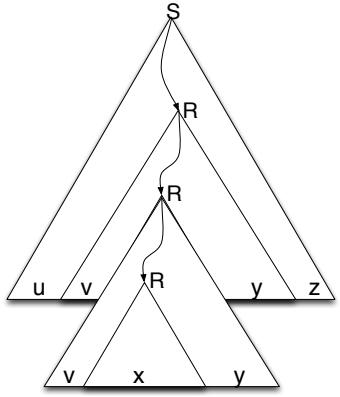
   (a) If a non-terminal is repeated on some path from the root to the leaves of a parse tree for a sentence of a language, then we can increase or decrease the number of occurrences of the symbol on the path leading to the duplication of substrings as required in the Lemma,

   (b) If a parse tree is sufficiently large, it must have repeated symbols on some path.

6. Suppose that some variable R appears more than once on a path from the root to some leaf of a parse tree relative to a CFG as shown in the figure below.

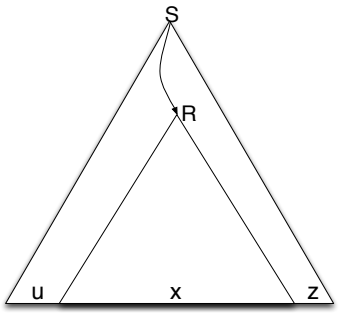


- In this case, we could replace the subtree rooted at the second occurrence of R with a copy of the full tree rooted at the first occurrence of R as shown below:



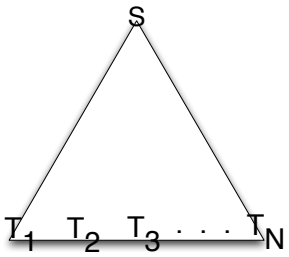
- If (as shown in the figure), x was the string at the frontier of the subtree rooted at the second occurrence of R, vxy was the string on the frontier of the subtree rooted at the first R, and $uvxyz$ was the frontier of the full parse tree, this shows that we could construct a parse tree for uv^2xy^2z showing that this string must be in the language of the grammar.
- Simply repeating the process of replacing the last smallest subtree rooted at R with the subtree rooted at the preceding copy of R shows that uv^ixy^iz is in the language for all positive values of i.
- Similarly, replacing the subtree rooted at the first occurrence of

R with the subtree rooted at the second shows that uv^0xy^0z must be in the language.

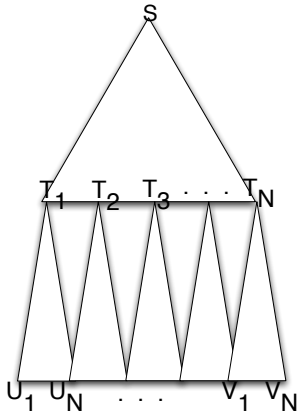


7. All we need to show to prove that sufficiently large strings in a CFL can be pumped is that some variable must repeat along a path from the root to the leaves of the parse tree of any such string.

   - Given a CFL C we know that we can find some CFG $G = (V, \Sigma, R, S)$ such that $L(G) = C$.
   - Suppose that N is the largest number of symbols in the right-hand side of any rule in R.
   - Consider how the height of a parse tree is related to the length of the associated string w:
     - If the parse tree has height 1, then it has at most N leaves:



     - If the parse tree has height 2, then it has at most N^2 leaves since each of the N symbols one step from the root can have produced at most N leaves.



     - In general, therefore, a tree of height H can have at most N^H leaves. In other words, if $w \in L(G)$ and $|w| \geq N^H$, then the parse tree for w must have some path containing more than H interior nodes.
     - Suppose that we choose w such that $|w| \geq N^{|V|+1}$. Then the parse tree for w must contain some path of length greater than $|V|$. That is, there must be more nodes in the path then there are variables in the grammar.
     - Just as in the proof of the Pumping Lemma for regular languages, we observe that the sequence of variables that label a path of length greater the number of variables must contain at least one repeated variable.

8. This is how we can determine a bound on the value of the pumping length p described in the Pumping Lemma. If $p = N^{|V|+1}$ then the parse tree for any string of length p or greater will contain a path with a repeated variable and therefore the string can be pumped.

9. To ensure that the vxy we are pumping satisfy all of the conditions of the Pumping Lemma, we have to be a bit careful. Rather than picking any parse tree for w, we have to pick the smallest one. This ensures we cannot have used a rule like $R \to R$ on the path we select (since otherwise we could get a smaller parse tree by removing it). This means that either v or y must be non-empty. Also, if there are

multiple repeated variables, we must pick a pair as close to the leaves as possible to ensure that $|vxy| < p$.

Using the Pumping Lemma for Context-free Languages

(Click for video)

1. As a very simple example of how we could use this lemma, consider $L = \{0^n1^n0^n \mid n \geq 0\}$:
   - Consider the string $0^p1^p0^p = uvxyz$ with $|vxy| \leq p$.
   - Since vxy is no more than one third the length of $0^p1^p0^p$ it can overlap no more than two of the three segments of 0's and 1's that make up the string.
   - Therefore, in the string uv^2xy^2z the lengths of at least one of the segments of 0's and 1's will remain p while either one or two of the other segments will increase in length.
   - Therefore, uv^2xy^2z cannot be of the form $0^n1^n0^n$ for any n. If the language was context-free, this would contradict the pumping lemma. Therefore, the language must not be context-free.

Deciding whether a language is a CFL?

1. NO VIDEO????
2. The following section of notes cover several more examples of the Pumping Lemma for CFL's. Given that we had an extra week of spring break (and wasn't it fun?!), I have to find some material to skip over the next few weeks. These examples are one of the things I am going to skip (in the sense that there will be no video you have to watch), but I thought I would still include the notes in case you wanted to look at some of the examples.
3. Let's consider a few problems that should provide some practice for deciding whether or not a language is context-free. For each language described, try to either think of a string in the language that cannot be pumped or described a grammar or PDA for the language:
   - Is $L = \{a^nb^ja^nb^j \mid j, n \geq 0\}$ a CFL?
   - How about $C = \{a^nww^Ra^n \mid n \geq 0, w \in \{a, b\}^*\}$?
   - How about $C' = \{a^nwa^nw^R \mid n \geq 0, w \in \{a, b\}^*\}$?
   - How about notP = $\overline{\{w \mid w \in \{a, b\}^* \text{ and w is a palindrome}\}}$
4. We can show that the first is not a CFL, the second is, and the third...
   - Consider whether $L = \{a^nb^ja^nb^j \mid j, n \geq 0\}$ is a CFL?

     If C is a CFL, then there is some p such that all strings longer than p can be pumped. Consider the string $w = a^pb^pa^pb^p$ and consider whether w can be partitioned as $w = uvxyz$ in any way that satisfies the requirements of the pumping lemma.
     First, note that all members of L are also members of $a^*b^*a^*b^*$. That is, each string in L contains at most 2 alternating pairs of a strings of as followed by a string of b. If w is partitioned in such a way that either v or y contained both as and bs, then $w' = uv^2xy^2z$ would no longer be a member of $a^*b^*a^*b^*$ since it would contain at least three alternating pairs of as and bs. Therefore both v and y must fall entirely within a sequence of as or a sequence of bs.
     Given that v falls in (including on the trailing edge of) a given strings of as or bs, y must either fall entirely in the same group or in the following group. This means that in the string $w' = uv^2xy^2z$, the length of at least one and no more than two of the four groups of substrings of the same symbol in w will have grown while at least two will remain unchanged. Moreover, if two of the substrings have changed, they must be adjacent. Thus, if the first strings of as is longer in w' the second strings of as will not have grown and therefore cannot match it. Similarly, if the second substring (the first string of bs) is the first substring whose length has increased, the final substring of bs will not have changed. Finally, if v falls in the second half of w, then one of the two substrings in the second

half of w' will have grown and therefore no longer match it partner in the first half.

- How about $C = \{a^n ww^R a^n \mid n \geq 0, w \in \{a,b\}^*\}$?

  First, note that we can pump any string of this form. If w is empty this is obvious. If w is non-empty, then let u be the prefix up until the last symbol of w, let $x = \epsilon$, $v = y =$ the last character of w (which is also the first character of w^R and let z be the rest of the string. Pumping by setting $i > 1$ just adds multiple copies of the last character of w to w and its reverse so the string remains within the language. Setting $i = 0$ just deletes the last character.

  The fact that a language satisfies the Pumping Lemma does not prove it is context-free. To accomplish this we need to show that we can describe the language with a PDA or context-free grammar.

  We know that the grammar

  $$P \to aPa \mid \epsilon$$

  generates $\{a^n a^n \mid n \geq 0\}$ and that

  $$S \to aSa \mid bSb \mid \epsilon$$

  generates $\{ww^R \mid w \in \{a,b\}^*\}$ We can combine these to form:

  $P \to aPa \mid S$
  $S \to aSa \mid bSb \mid \epsilon$

  which will generate C. Even easier, we can note that

  $$\{a^n ww^R a^n \mid n \geq 0, w \in \{a,b\}^*\} = \{ww^R \mid w \in \{a,b\}^*\}$$

  and just use the second grammar.

- Consider the string $a^p b^p a^p b^p \in C' = \{a^n wa^n w^R \mid n \geq 0, w \in \{a,b\}^*\}$. For any $uvxyz = a^p b^p a^p b^p$, with $|vxy| \leq p$ either vxy is completely contained in one of the 4 specified subsections of length p in w or it overlaps two of them in which case it cannot overlap the first character of the first quarter or the last character of the second quarter.

  Let's start by seeing that this isn't as simple as it first appears. Suppose vxy falls entirely within the second a^p. It is tempting to naively say that in that case $uv^i xy^i z$ cannot be in C' since it would have too many as in the third component to match the as in the first component. However, if the number of as in the combination of v and y is even (say $2k$), we can view half of them as belonging to w which grows from b^p to be $b^p a^k$ and the other half as belonging to w^R making it $a^k b^p$ so that the total string is viewed as $a^p\ b^p a^k\ a^p\ a^k b^p$.

  Luckily, if we pump down removing a's from the third component, there is no way we can remain within the language since there won't be enough a's anywhere in the last 3/4 to match the first a^p.

  Pumping within any of the other quarters is less subtly impossible. If you increase the size of the first collection of a's you cannot treat some of them as part of the reversed w because the extras would have to show up at the end of the string. Increasing either set of b's alone will not work because reversal does not change the number of b's leading to an unavoidable mistmatch.

  and so on...

- What language is described by:

  $R \to XRX \mid S$
  $S \to aTb \mid bTa$
  $T \to XTX \mid X \mid \epsilon$
  $X \to b \mid a$

  Answer: The complement of the set of palindromes over $\{a,b\}$.

  It is worth noting, however, that this is an exception to the rule. In general, context-free languages are not closed under complement.

Closure Properties of CFLs and Deterministic CFLs

(Click for video)

1. Recall that the regular languages were closed under many operations: complement, union, intersection, closures, ...

2. The context-free languages don't have nearly as many nice closure properties

   - Context-free languages are closed under union. Given two context-free grammars, $G_1 = (V_1, \Sigma, S_1, R_1)$ and $G_1 = (V_2, \Sigma, S_2, R_2)$,[1] it is easy to build a grammar for their union by introducing a new start symbol that can derive either of the start symbols of the original languages. That is, the new grammar is built by adding a new start symbol S with rules $S \to S_1$ and $S \to S_2$. Formally,

     $$G_{union} = (V_1 \cup V_2 \cup \{S\}, \Sigma, S, R_1 \cup R_2 \cup \{(S, S_1), (S, S_2)\})$$

   - Similarly, they are closed under product. Given two context-free grammars, $G_1 = (V_1, \Sigma, S_1, R_1)$ and $G_1 = (V_2, \Sigma, S_2, R_2)$, it is easy to build a grammar for their product by adding a new start symbol S and the rule $S \to S_1 S_2$ to the rules of the original grammars. Formally,

     $$G_{product} = (V_1 \cup V_2 \cup \{S\}, \Sigma, S, R_1 \cup R_2 \cup \{(S, S_1 S_2)\})$$

3. Context free languages are not closed under intersection.

   - Consider the languages $L_1 = \{0^n 1^n 0^m \mid n, m \geq 0\}$ and $L_2 = \{0^m 1^n 0^n \mid n, m \geq 0\}$. Both of these languages are context-free. We can argue this easily using some of the closure properties just discussed:
     - A grammar with the two rules $S \to 0S1$ and $S \to \epsilon$ describes the language $\{0^n 1^n \mid n \geq 0\}$, so this language is context-free.
     - The language described by the regular expression 0^* must be context-free since all regular languages are context free.
     - L_1 is the product of these two languages so it must be context-free.
     - A similar argument can be provided for L_2.
   - The intersection of L_1 and L_2 is $\{0^n 1^n 0^n \mid n \geq 0\}$ which we have already shown is not context-free (at the end of our discussion of the Pumping Lemma).

4. This implies that the complement of a context-free language is not necessarily context-free.

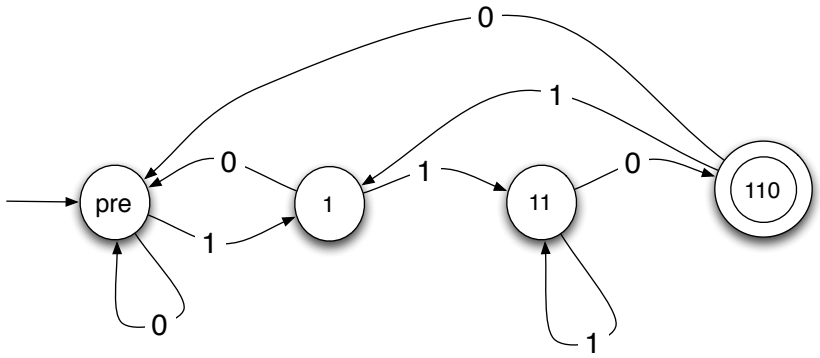
   - Recall that for any sets A and B, $A \cap B = \overline{\overline{A} \cup \overline{B}}$.
   - As a result of this fact, if context-free languages were closed under complement and union, they would need to be closed under intersection.
   - We have just seen that context-free languages are closed under union but not under intersection.
   - This implies that context-free languages cannot be closed under complement.

5. The fact that context-free languages are not closed under complement has an important consequence. It shows that non-determinism is a significant feature of push-down automata.

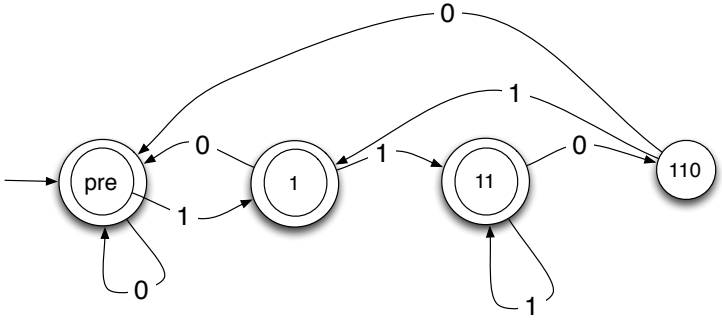
6. Recall how we were able to show that regular languages were closed under complement.

   - Hint: We didn't do it using regular expressions!
   - Our proof was based on a simple property of deterministic finite automata. If you complement the set of final states of a deterministic finite automaton, you complement the language accepted.
     - For example, if you take the machine:

[1] Where, for simplicity we assume that the grammars have disjoint sets of variable/non-terminal names and identical terminal alphabets.



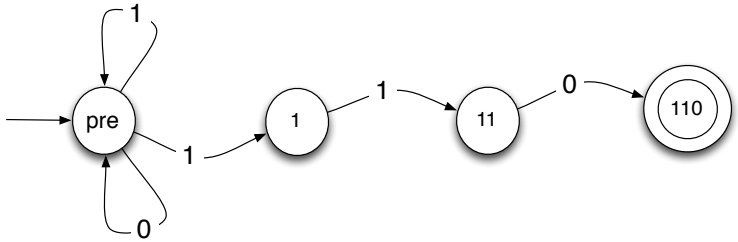
which recognizes strings that end in 110, and make its final states non-final and vice versa, you get the machine below



which recognizes strings that don't end in 110.

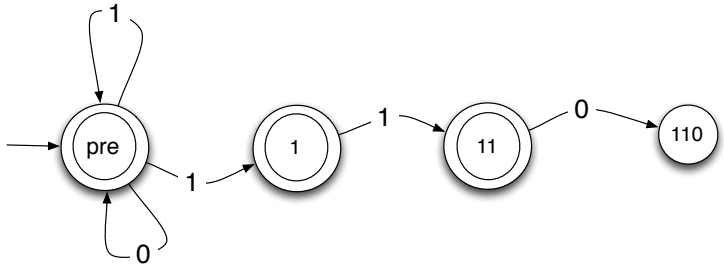
7. Now, consider how this trick plays out if we apply it to a non-deterministic finite automaton.

   - Consider the machine



   which recognizes strings that end in 110. We used this machine weeks ago to motivate the advantages of non-determinism since its structure makes it much clearer what it does that the structure of the corresponding DFA.

   - If we interchange the final and non-final states of this machine we get



   which recognizes Σ^*! This is clearly not the complement of the language of the other machine.

8. Given a deterministic PDA, i.e., a PDA $D = (Q, \Sigma, \Gamma, \delta, s, F)$, where $|\delta(q, x)| = |x|, x \in \Sigma_\epsilon$, it is the case that if $\overline{D} = (Q, \Sigma, \Gamma, \delta, s, Q - F)$, then $L(\overline{D}) = \overline{L(D)}$.

9. We call a language that is recognized by a deterministic PDA a deterministic context-free language. DCFLs are closed under complement. Therefore, the set of DCFLs is a strict subset of the set of CFLs.

Chomsky Normal Form

(Click for video)

1. Have you ever heard of Noam Chomsky?

2. He is best know these days for his political activism.

3. Before his political career, Chomsky played a significant role in the development of formal grammars (of which CFGs are one example).

4. One of his contributions was describing a restricted form of CFG that was powerful enough to describe any CFL. In particular, we say that a

grammar is in Chomsky Normal Form, if all productions in the grammar are of the forms:

- $A \to BC$, where A, B, and C are all non-terminals,
- $A \to t$ for some non-terminal A and terminal t, or
- $S \to \epsilon$ where S is the start symbol.

5. At some level, this is not all that surprising. In particular, if you take a "typical" production like

$$S \to if(B)\ S$$

it can be replaced by a collection of productions of the second type to handle the terminals:

$T_1 \to i$
$T_2 \to f$
$T_3 \to ($
$T_4 \to)$

together with a collection of non-terminals and chain of rules to capture the original rule:

$S \to T_1 S_1$
$S_1 \to T_2 S_2$
$S_2 \to T_3 S_3$
$S_3 \to B S_4$
$S_4 \to T_4 S$

6. The tricky part is finding a replacement for productions like $T \to \epsilon$ and $T \to N$.

- The basic idea is that if such rules are present, we will make multiple copies of other productions that reference the variable on the left side (T in our examples) with multiple productions that reflect the option of using these simple rules.
- For example, if our grammar included the rules

$S \to if(B)\ S$
$S \to if(B)\ S\ else\ S$
$S \to \epsilon$
$S \to T$

  - We would begin by adding a copy of the first rule that reflected the possibility of using the ϵ-rule:

    $S \to if(B)$

  - This is a bit trickier for the second rule because we have to account for the fact that either one, both, or neither of the instance of S in the rule might be expanded using the ϵ-rule:

    $S \to if(B)\ else\ S$
    $S \to if(B)\ S\ else$
    $S \to if(B)\ else$

  - Similar productions would be added to take the place of the $S \to N$ rule:

    $S \to if(B)\ N$
    $S \to if(B)\ N\ else\ S$
    $S \to if(B)\ S\ else\ N$
    $S \to if(B)\ N\ else\ N$

  - After all these additions are made, the offending rules can be removed

7. The existence of Chomsky Normal Form means that if we are given a context-free grammar and a string we can determine whether the string belongs to the language of the grammar.

- You may already think this is obvious because we could just build a PDA from the grammar for the language, but PDAs are non-deterministic and simulating the non-determinism to see if there is any combination of production choices that leads to the desired string might go on forever if no such choices existed.
- Chomsky Normal form means we can find a grammar for the language with the property that the sentential forms we generate get larger and larger as we do more production steps.
- This means we can cut off the non-deterministic search once we

reach any sentential form that is longer than the desired sentence and backtrack to explore other short derivations.