CS 3100, Models of Computation, Spring 20, Lec15

Ganesh Gopalakrishnan
School of Computing
University of Utah
Salt Lake City, UT 84112

URL: bit.ly/3100s20Syllabus

SCHOOL OF COMPUTING
THE UNIVERSITY OF UTAH

Recap

- DFA → Purely right-linear
- Take purely right-linear productions and reverse each rule to obtain a purely left-linear production system
  - By doing so, we would have reversed the language of a DFA
  - This result (after reversal) is also regular
  - Thus we can argue that purely left-linear productions also denote regular languages
- Thus, purely right-linear and purely left-linear productions denote regular languages
- With mixed linearity, we don’t have this guarantee
  - S -> (A | “
  - A -> S) | “

Recap

- Consistency:
  - Are we producing ONLY legal strings?
  - S -> a S b S | ‘’ as a single rule is consistent (it is not complete)
  - Proof by induction:
    - Assume that the RHS “S” are consistent; showthat the LHS “S” are consistent
- Completeness:
  - Are we producing ALL legal strings?
  - S -> a S b S | b S a S | ‘’ makes it complete for “equal a’s and b’s”
    - Assume that all strings of length <= N and in the language of interest are derivable. Show that the next eligible length of stings are derivable
- Proof by visualizing the strings
  - “hill/valley” plots

Which are CFL and which aren't? (intuitively)

1. $L_{P0} = \{w \ : \ w \in \Sigma^*\}$
2. $L_{P1} = \{ww^R \ : \ w \in \Sigma^*\}$
3. $L_{P2} = \{waw^R \ : \ a \in (\{\varepsilon\} \cup \Sigma), \ w \in \Sigma^*\}$
4. $L_{eq01} = \{0^n 1^n \ : \ n \geq 0\}$
5. $L_{ww} = \{ww \ : \ w \in \Sigma^*\}$
6. $L_{w\#w} = \{w\#w \ : \ w \in \Sigma^*\}$, where # is a separator.
7. $L_{eq010} = \{0^n 1^n 0^n \ : \ n \geq 0\}$
8. $L_{eq012} = \{0^n 1^n 2^n \ : \ n \geq 0\}$

How to prove that a language is NOT a CFL?

- We have a Pumping Lemma for CFLs!
- Used to show that a given language is not context-free
- Usage similar to the regular-language pumping lemma
- The “pump” happens for a different reason
  - Long strings have tall parse trees
  - In any tall parse tree, some nonterminal repeats along a tree path
  - This gives us the opportunity to generate LONGER or SHORTER strings

Consider this CFG… let's show the pump.

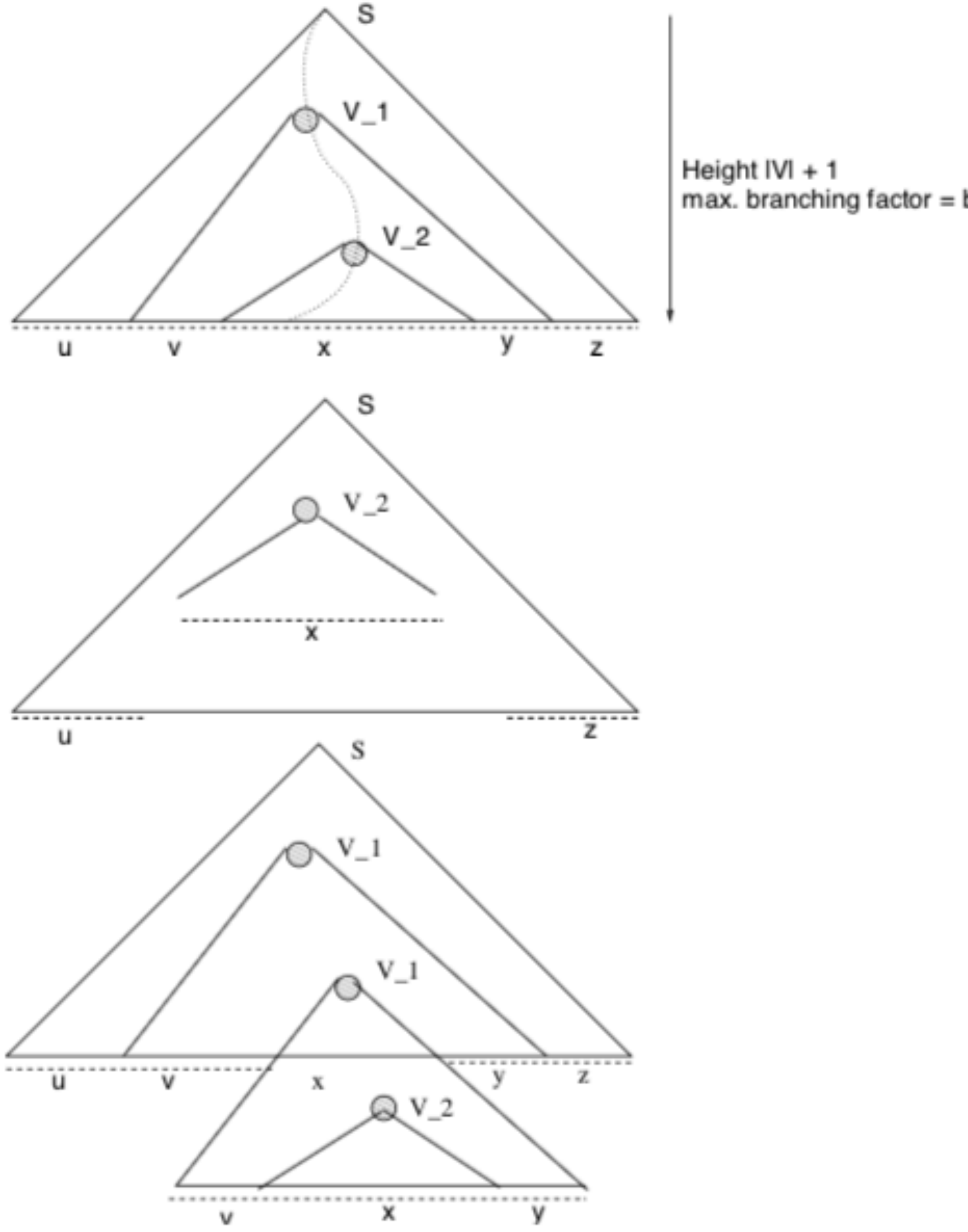
```
S -> ( S ) | T | ''
T -> [ T ] | T T | ''
```

Summary of Example

```
Given that this              We infer that this            OR, this
derivation exists:           derivation exists:            derivation exists:
===================          ===================           ===================
S => (S)                     S => (S)                      S => (S)
  => (( T ))                   => (( T ))                    => (( T ))
  => (( [ T ] ))               => (( [ T ] ))                => ((   ))
  => (( [   ] ))               => (( [[ T ]] ))
                               => (( [[[ T ]]] ))
                               => ...
                               => (( [[[[[[[[ T ]]]]]]]] ))
                               => (( [[[[[[[[   ]]]]]]]] ))
```

```
S -> ( S ) | T | ''
T -> [ T ] | T T | ''
```

CFL PL in Pictures



The CFL PL finally! (pictures)

Theorem 11.9: Given any CFG $G = (N, \Sigma, P, S)$, there exists a number p such that given a string w in $L(G)$ such that $|w| \geq p$, we can split w into $w = uvxyz$ such that $|vy| > 0$, $|vxy| \leq p$, and for every $i \geq 0$, $uv^ixy^iz \in L(G)$.

The CFL PL finally! (words)

- Suppose L_{ww} were a CFL.
- Then the CFL Pumping Lemma would apply.
- Let p be the pumping length associated with a CFG of this language.
- Consider the string $0^p1^p0^p1^p$ which is in L_{ww}.
- The segments v and y of the Pumping Lemma are contained within the first 0^p1^p block, in the middle 1^p0^p block or in the last 0^p1^p block, and in each of these cases, it could also have fallen entirely within a 0^p block or a 1^p block.
- In each case, by pumping up or down, we will then obtain a string that is not within L_{ww}. $\square$

Context-free Grammars (CFG)

A context-free grammar is a four-tuple (N, Σ, S, P), where

- N is a set of **nonterminals**. In L_{Dyck}, `S` is the only nonterminal.
- Σ is a set of **terminals**. In L_{Dyck}, the terminals are `(` and `)`. The name "terminals" suggests places when the recursion of the context-free production rules *terminates*. ε itself can be viewed as a terminal, although strictly speaking, it is not. When we define P below, we will allow the right-hand sides of production rules to contain $\{(N \cup \Sigma)^*$. From that point of view, ε (ASCII `''`) is an *empty string of terminals*.
- S is the **start symbol** which is one of the nonterminals. In our example, the start symbol is `S`.
- P is a set of **production rules** which are of the form:
  - $N \to \{(N \cup \Sigma)^*$, and read "`N` derives a string of other `N` and Σ items." Such strings are called **sentential forms**. A terminal-only string is called a **sentence**.

A complete exercise

- Show that

{ w w : w in Sigma* } is not a CFL but its complement is a CFL

An exercise on consistency and completeness

Consider the CFG

S -> (W S | ''

W -> (W W |)

What language does S generate? Prove via Consis. Compl.

- Consistency: State the consistency goals of S and W
  - Prove the consistency of S and W
- Completenes: State the completeness goals of S and W
  - Prove the completeness goals of S and W

The Makeup Midterm question

#1 > #0

Draw hill-valley plots

Dissect the plot

Express as a CFG!