MA/CSSE 474
Theory of Computation

Pumping Theorem Examples

From: Seth Troisi [mailto:braintwo@gmail.com]
Sent: Friday, April 20, 2018 1:52 PM
To: Anderson, Claude W <anderson@rose-hulman.edu>
Subject: Theory of computation

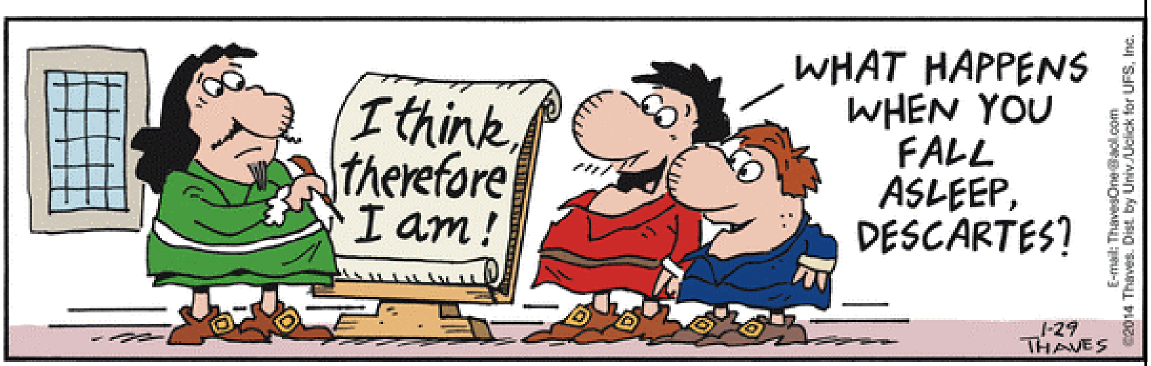
You taught one of my favorite class (CS474) at all of Rose. Building DFA, PDA, and Turing Machine was lots of fun!

This year's Google CodeJam final had a question about the size of the language accepted by a various regular expressions. I helped prepare the internal solution and analysis for the problem.

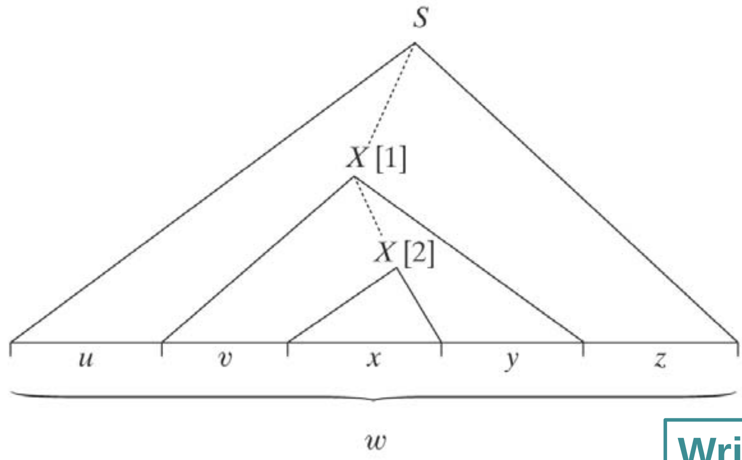
Just reaching out again to say how much fun math is,
Seth

Your Questions?

- Previous class days' material
- Reading Assignments
- HW 12 or 13 problems
- Anything else



The Context-Free Pumping Theorem



Write it in contrapositive form. Try to do this before going on.

If L is a context-free language, then
$\exists k \geq 1$ ($\forall$ strings $w \in L$, where $|w| \geq k$
($\exists u, v, x, y, z$ ($w = uvxyz$,
$vy \neq \varepsilon$,
$|vxy| \leq k$, and
$\forall q \geq 0$ (uv^qxy^qz is in L)))).

Pumping Theorem contrapositive

- We want to write it in contrapositive form, so we can use it to show a language is NOT context-free. **Original:**

If L is a context-free language, then
$\exists k \geq 1$ ($\forall$ strings $w \in L$, where $|w| \geq k$
($\exists u, v, x, y, z$ ($w = uvxyz$,
$vy \neq \varepsilon$,
$|vxy| \leq k$, and
$\forall q \geq 0$ (uv^qxy^qz is in L)))).

Contrapositive: If
$\forall k \geq 1$ ($\exists$ string $w \in L$, where $|w| \geq k$
($\forall u, v, x, y, z$
($w = uvxyz$,
$vy \neq \varepsilon$,
$|vxy| \leq k$, and
$\exists q \geq 0$ (uv^qxy^qz is not in L)))),
then L is not a CFL.

Regular *vs*. CF Pumping Theorems

Similarities:

- We don't get to choose k.
- We choose w, the string to be pumped, based on k.
- We don't get to choose how w is broken up (into xyz or uvxyz)
- We choose a value for q that shows that w isn't pumpable.
- We may apply closure theorems before we start.

Things that are different in CFL Pumping Theorem:

- Two regions, v and y, must be pumped in tandem.
- We don't know anything about where in the strings v and y will fall in the string w. All we know is that they are reasonably "close together", i.e., $|vxy| \leq k$.
- Either v or y may be empty, but not both.

An Example of Pumping: $A^nB^nC^n$

$A^nB^nC^n = \{a^nb^nc^n, n \geq 0\}$

Choose $w = a^k b^k c^k$ *(we don't get to choose the k)*
1 | 2 | 3 (the regions: all a's, all b's, all c's)

If either v or y spans two regions, then let q = 2 (i.e., pump in once). The resulting string will have letters out of order and thus not be in $A^nB^nC^n$.

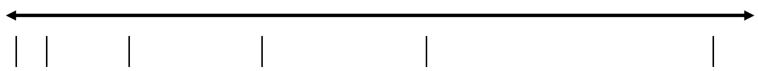
Other possibilities for (v region, y region)
(1, 1): q=2 gives us more a's than b's or c's. (2, 2) and (3,3) similar.
(1, 2): q=2 gives more a's and b's than c's. (2, 3) is similar.
(1, 3): Impossible because |vxy| must be ≤ k.

An Example of Pumping: { a^{n^2}, $n \geq 0$}

$L = \{a^{n^2}, n \geq 0\}$

The elements of L:

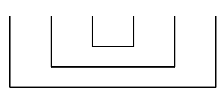
n	w
0	ε
1	a^1
2	a^4
3	a^9
4	a^{16}
5	a^{25}
6	a^{36}



Nested and Cross-Serial Dependencies

PalEven = $\{ww^R : w \in \{a, b\}^*\}$

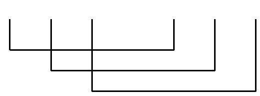
a a b b a a



The dependencies are nested. Context-free.

WcW = $\{wcw : w \in \{a, b\}^*\}$

a a b c a a b



Cross-serial dependencies. Not context-free.

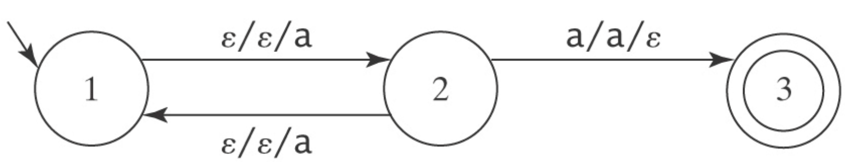
Work with one or two other students on these

- **{$a^n b^m a^n$, n, m ≥ 0 and n ≥ m}**
- **WcW = {wcw : w ∈ {a, b}*}**
- **{$(ab)^n a^n b^n$: *n* > 0}**
- **{*xcy* : *x*, *y* ∈ {0, 1}* and *x* ≠ *y*}**

Halting

It is possible that a PDA may
- not halt,
- never finish reading its input.

Let Σ = {a} and consider *M* =



L(*M*) = {a}: (1, a, ε) |- (2, a, a) |- (3, ε, ε)

On any other input except a:
- *M* will never halt.
- *M* will never finish reading its input unless its input is ε.

Nondeterminism and Decisions

1. There are context-free languages for which no deterministic PDA exists.

2. It is possible that a PDA may
   - not halt,
   - not ever finish reading its input.
   - require time that is exponential in the length of its input.

3. There is no PDA minimization algorithm.
   It is undecidable whether a PDA is minimal.

Solutions to the Problem

- For NDFSMs:
  - Convert to deterministic, or
  - Simulate all paths in parallel.

- For NDPDAs:
  - No general solution.
  - Formal solutions usually involve changing the grammar.
    - Such as Chomsky or Greibach Normal form.
  - Practical solutions:
    - Preserve the structure of the grammar, but
    - Only work on a subset of the CFLs.
      - LL(k), LR(k)   (compilers course)

Closure Theorems for Context-Free Languages

The context-free languages are closed under:

- Union
- Concatenation
- Kleene star
- Reverse

Let $G_1 = (V_1, \Sigma_1, R_1, S_1)$, and
$G_2 = (V_2, \Sigma_2, R_2, S_2)$
generate languages L_1 and L_2

Formal details on next slides;
we will do them informally

Closure Under Union

Let $G_1 = (V_1, \Sigma_1, R_1, S_1)$, and
$G_2 = (V_2, \Sigma_2, R_2, S_2)$.

Assume that G_1 and G_2 have disjoint sets of nonterminals, not including S.

Let $L = L(G_1) \cup L(G_2)$.

We can show that L is CF by exhibiting a CFG for it:

$G = (V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2,$
$R_1 \cup R_2 \cup \{S \rightarrow S_1, S \rightarrow S_2\},$
$S)$

Closure Under Concatenation

Let $G_1 = (V_1, \Sigma_1, R_1, S_1)$, and
$G_2 = (V_2, \Sigma_2, R_2, S_2)$.

Assume that G_1 and G_2 have disjoint sets of nonterminals, not including S.

Let $L = L(G_1)L(G_2)$.

We can show that L is CF by exhibiting a CFG for it:

$$G = (V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, \\ R_1 \cup R_2 \cup \{S \to S_1\ S_2\}, \\ S)$$

Closure Under Kleene Star

Let $G = (V, \Sigma, R, S_1)$.

Assume that G does not have the nonterminal S.

Let $L = L(G)^*$.

We can show that L is CF by exhibiting a CFG for it:

$$G = (V_1 \cup \{S\}, \Sigma_1, \\ R_1 \cup \{S \to \varepsilon, S \to S\ S_1\}, \\ S)$$

Closure Under Reverse

$L^R = \{w \in \Sigma^* : w = x^R \text{ for some } x \in L\}$.

Let $G = (V, \Sigma, R, S)$ be in Chomsky normal form.

Every rule in G is of the form $X \rightarrow BC$ or $X \rightarrow a$, where X, B, and C are elements of $V - \Sigma$ and $a \in \Sigma$.

- $X \rightarrow a$: $L(X) = \{a\}$. $\{a\}^R = \{a\}$.
- $X \rightarrow BC$: $L(X) = L(B)L(C)$. $(L(B)L(C))^R = L(C)^R L(B)^R$.

Construct, from G, a new grammar G', such that $L(G') = L^R$:
$G' = (V_G, \Sigma_G, R', S_G)$, where R' is constructed as follows:

- For every rule in G of the form $X \rightarrow BC$, add to R' the rule $X \rightarrow CB$.
- For every rule in G of the form $X \rightarrow a$, add to R' the rule $X \rightarrow a$.