Lecture 12: Closure operations and algorithms for CFLs

CSCI 101
Spring, 2019

Kim Bruce

Closure properties of CFL's

- Already shown closed under
  - concatenation, union, Kleene*, reversal, substitution
- Also closed under intersection with regular set.
  - Product machine
- Not closed under intersection, difference, or complement.
  - Why doesn't product work for intersection?

Counter Example

- Let
  - $L_1 = \{a^n b^n c^m \mid m, n \geq 0\}$, clearly cfl
  - $L_2 = \{a^m b^n c^n \mid m, n \geq 0\}$, clearly cfl
  - $L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 0\}$ is not cfl

CFL's not closed under intersection

- Spose cfl's closed under complement
  - I.e., if L is cfl then $L^c = \Sigma^* - L$ is cfl.
  - Then $L_1 \cap L_2 = (L_1^c \cup L_2^c)^c$ would be cfl. Contradiction!

CFL's not closed under complement

- CFL's not closed under complement, difference, or intersection.
  - If closed under difference then would be closed under complement!
- Closure can be used to prove languages not cfl.
  - Spose $L = \{w \mid \#_a(w) = \#_b(w) = \#_c(w)\}$ is cfl
  - Then $L \cap a^*b^*c^* = \{a^n b^n c^n \mid n \geq 0\}$, which is not cfl
  - Because cfl's closed under $\cap$ w/ regular set, L not cfl

Algorithms for CFL's

- Given a cfg, G, and w in Σ^*, is $w \in L(G)$?
- Given a cfg, G, is $L(G) = \emptyset$?
- Given a cfg, G, is L(G) infinite?
- All are decidable!

Is $w \in L(G)$?

- Special case if $w = \varepsilon$, *see next slide*.
- Convert G to CNF G'
- If $|w| = n$ look at all derivations of length $\leq 2\,|w| - 1$. If not there, then not in language.
- How efficient? Let $|w| = n$
  - $|R|^{2n-1}$ derivations, each of length $2n-1$. Thus $O(n\ 2^n)$
  - With work (see later), can find $O(n^3)$ algorithm.
    - Want O(n)!!

Is ε in L(G)?

- Say A is nullable iff $A \Rightarrow^* \varepsilon$.
- Lemma: A is nullable iff
  - $A \rightarrow \varepsilon$ or
  - $A \rightarrow B_1 \ldots B_n$ and all B_i are nullable
- To see if ε in L(G), see if S is nullable.

Algorithms for CFL's

- Given a cfg, G, and w in Σ^*, is $w \in L(G)$? ✔
- Given a cfg, G, is $L(G) = \emptyset$?
- Given a cfg, G, is L(G) infinite?
- All are decidable!

Thinning cfg

- Say non-terminal V is non-productive if there is no string $w \in \Sigma^*$ s.t. $V \Rightarrow^* w$.
- Algorithm: Start with G' = G
  - Mark every terminal in G' as productive
  - Until entire pass through R w/no marking
    - For each $X \rightarrow \alpha$ in R: If every symbol in α marked as productive, but X not yet marked productive, then mark it as productive
  - Remove from $V_{G'}$ all non-productive symbols
  - Remove from $R_{G'}$ all rules w/non-productive symbols on left or right side.

Algorithm for Emptiness

- Run algorithm to mark non-productive symbols. If S non-productive then $L(G) = \emptyset$.

Algorithm for Finiteness

- Let G be cfg. Use proof of pumping lemma.
  - Let G' be equivalent grammar in CNF.
  - Let n = #non-terminals. Let $k = 2^{n+1}$.
  - If there is a $w \in L(G)$ s.t. $|w| > k$ then can pump, so ∞
  - Claim if L(G) ∞ then exist $w \in L(G)$ s.t. $k < |w| \leq 2k$.
    - Spose fails. Then ∞, so let $w' \in L(G)$ be shortest s.t. $|w'| > 2k$.
    - Pump with i = 0 to get shorter. But $|vxy| < k$ & thus $|vy| < k$.
    - Thus $uxz \in L(G)$, $|uxz| < |w'|$, but $|uxz| > k$. Contradiction to assumption w' shortest!
    - Thus L(G) is ∞ iff exists $w \in L(G)$ s.t. $k < |w| \leq 2k$

Parsing CFL's

- Created non-deterministic PDA.
  - Backtracking computation hard to get right.
  - Having to backtrack on input painful
- More efficient dynamic programming
- Later see deterministic language better

Compiler Overview

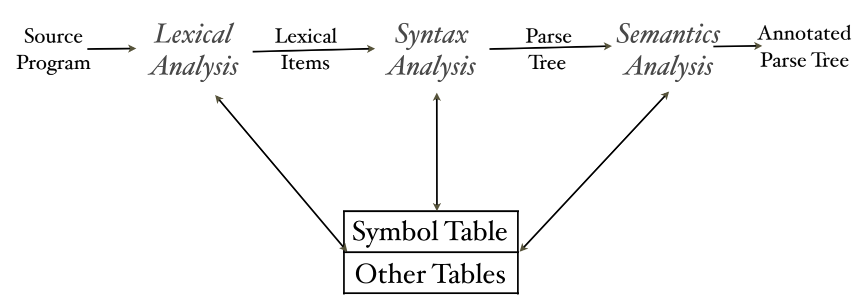
Compiler Structure

- Analysis:
  - Break into lexical items, build parse tree, annotate parse tree (e.g. via type checking)
- Synthesis:
  - generate simple intermediate code, optimization (look at instructions in context), code generation, linking and loading.

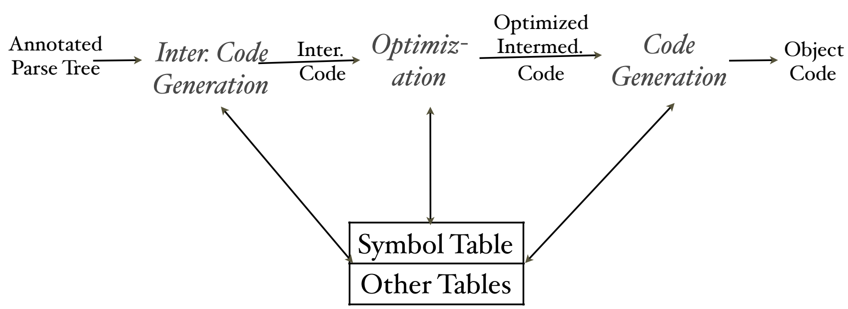
Symbol Table

- Symbol table:
  - Contains all id names,
  - kind of id (vble, array name, proc name, formal parameter),
  - type of value,
  - where visible, etc.

Analysis



Synthesis



Focus

- Assume lexical analysis done
  - Described by regular expressions
  - Recognized by DFSM
  - Resulting input is stream of tokens
- Focus on building parse trees

Parsing CFL's

- Created non-deterministic PDA.
  - Backtracking computation hard to get right.
  - Having to backtrack on input painful
- More efficient via dynamic programming
- Deterministic language better as can get linear recognizer.
  - LR(1) and LL(1) only require looking at next token of input.

CYK (for non-deterministic)

- Convert cfg G to G' in Chomsky Normal Form
- Let $w = w_1 \ldots w_n$ be string to be parsed. Define $\alpha(i,j)$ to be $\{B \mid B \Rightarrow^* w_i \ldots w_j\}$
  - So $w \in L(G)$ iff $S \in \alpha(1,n)$
  - Key idea: What non-terminals give substrings?
- Recursive definition:
  - $\alpha(i,i) = \{C \mid C \rightarrow w_i\}$
  - $\alpha(i,k) = \{C \mid C \rightarrow AB \ \&\ A \in \alpha(i,j) \wedge B \in \alpha(j+1,k)$ for some $j\}$

Fill in Table

$\alpha(1,1)$	$\alpha(1,2)$	$\alpha(1,3)$	…	$\alpha(1,n-1)$	$\alpha(1,n)$
	$\alpha(2,2)$	$\alpha(2,3)$	…	$\alpha(2,n-1)$	$\alpha(2,n)$
		$\alpha(3,3)$	…	$\alpha(3,n-1)$	$\alpha(3,n)$
			…	…	…
				$\alpha(n-1,n-1)$	$\alpha(n-1,n)$
					$\alpha(n,n)$

Each entry computed from entries in same row & column:
$\alpha(1,3)$ from $\alpha(1,1)$ & $\alpha(2,3)$, $\alpha(1,2)$ & $\alpha(3,3)$, etc.
Slide across row and down column.
Why $O(n^3)$?

Using CYK

- Let G be grammar for balanced parens in CNF:
  - $S \rightarrow SS$, $S \rightarrow LT$, $S \rightarrow LR$
  - $T \rightarrow SR$
  - $L \rightarrow ($, $R \rightarrow)$
- Parse ( ) (( ))
- Generally most entries are empty
- What if two entries in same slot?
  - Better to store rule rather than just left-hand side.

Predictive Parsing

- Want to be able to parse languages without backtracking (even done efficiently).
- Talked earlier about deterministic pda's.
  - No choice as to what to do at each step.
- Some languages seem deterministic, but don't quite work with that definition.
  - Book uses $L = a^* \cup \{a^n b^n \mid n \geq 0\}$
  - Empty stack when at end if no b's.

Deterministic CFL

- L is deterministic context-free iff L$ can be accepted by a deterministic pda.
  - Easy to show L deterministic cfl $\Rightarrow$ L cfl
    - Guess when about to read $, then read no more input
- Deterministic cfl's closed under complement.
  - Reversing accept easy, but also have to worry about stack not empty, etc.
  - More complex -- see text.