CS 3313
Foundations of Computing:

Context Free Grammars – Applications

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

© slides based on material from
Peter Linz book, Hopcroft, Narahari

Some more properties of CFGs and Applications of CFGs

- Deterministic Context Free languages
  - Accepted by Deterministic PDAs
- Why DCFLs….class of CFLs that lend themselves to efficient parsing.
- Are CFGs useful in applications other than parsing ?
  - Malware Detection Algorithms using CFGs to recognize malware

CFG Application: Computer Security

- Malware (Malicious Software): intrusive software that is designed to damage/hijack/destroy computer systems (software and hardware).
  - Viruses, Worms, Trojans, ransomware, spyware, adware…
- A particularly effective malware technique:
- Polymorphic Malware..real life Transformers!
  - Constantly changes its features to evade detection….adapts to encryption
  - evades simple pattern-matching detection
    - Simple pattern matching = Reg. Expr!!

Applying CFGs for Malware Detection

- An application of context free grammar for a virus detection system(VDS).
  - Traditional virus detection tools such as firewalls rely on a database scan.
- Traditional VDSs are not effective dealing with *polymorphic malware*
- Efficient tools can be made using concept of CFG (Sequitur)
- Algo:
  - Extract workflow of program's functions (program dependence graph!)
  - Apply traditional firewall to remove trivial malware
  - Construct CFG from work flow
  - Scan each function to capture complexity/pattern
  - Compare generated serial grammar to detect malware
- Read the notes, and watch the video
  - Discussion in next class

Deterministic PDAs and Deterministic Context Free Languages

- A deterministic PDA (DPDA) is one where we have exactly one choice of moves from any configuration/ID
- Languages accepted by DPDAs are Deterministic Context Free Languags (DCFLs)

- Why should we be interested in DCFLs (DPDA) ?
  - Practical applications…..parsers are deterministic

  - Syntax of programming languages can be specified using a subclass of DCFLs known as LR(k) (or LL(k)) grammars…
    - Andy will provide an overview of parsing and a construction of a compiler.
      - Interested students can choose to explore this project (in lieu of a later homework)

Deterministic Context Free Languages

- Theorem: DCFLs and CFLs are not equivalent.
  - There are languages that are accepted by a PDA but not by a DPDA.
- Example: $L = \{a^n b^n\} \cup \{a^n b^{2n}\}$
  - We have a PDA to accept this language
    - From start state it non-deterministically goes to a DPDA that accepts $\{a^n b^n\}$ or to a DPDA that accepts $\{a^n b^{2n}\}$
  - There is no DPDA to accept this language
    - Proof: in the textbook….
    - Basic Idea: if such a DPDA exists then we can construct a PDA to accept $\{a^n b^n c^n\}$ which is not a CFL – contradiction.
- Closure properties of DCFLs:
  - Not closed under Union, Intersection
  - Closed under complementation, inverse homomorphism

Why bother with DPDAs?

- Because DPDAs are deterministic,

  they can be simulated efficiently:

  - Keep track of top of stack
  - Store an **action/goto** table that says what operations to perform on the stack and what state to enter on each input/stack pair
  - Loop over the input, processing input/stack pairs until the automaton rejects or ends in an accepting state with all input consumed.

- If G is a deterministic CFG, and we can generate a DPDA for it, then we have an efficient parser !!
  - Not quite…can still take exponential time….
  - BUT can define subclasses of DPDAs which can produce efficient parsers IF they are allowed to *"look ahead"* k symbols
- Next…..brief overview of CFG Parsing