CS21 Decidability and Tractability
Lecture 9
January 25, 2023

Outline
- deterministic PDAs
- on to Turing Machines

Deterministic PDA
- A NPDA is a 6-tuple \((Q, \Sigma, \Gamma, \delta, q_0, F)\)
  where:
  - \(\delta: Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \rightarrow P(Q \times (\Gamma \cup \{\varepsilon\}))\) is a function called the transition function
  - A deterministic PDA has only one option at every step:
    - for every state \(q \in Q\), \(a \in \Sigma\), and \(t \in \Gamma\), exactly 1 element in \(\delta(q, a, t)\), or
    - exactly 1 element in \(\delta(q, \varepsilon, t)\), and \(\delta(q, a, t)\) empty for all \(a \in \Sigma\)

Example deterministic PDA
\[\Sigma = \{0, 1\}\]
\[\Gamma = \{0, 1, \$$\}\]
\[L = \{0^n1^n : n \geq 0\}\]
(unpictured transitions go to a “reject” state and stay there)

Deterministic PDA
- A technical detail:
  - we will give our deterministic machine the ability to detect end of input string
  - add special symbol \(\$$\) to alphabet
  - require input tape to contain \(x\$$
  - language recognized by a deterministic PDA is called a deterministic CFL (DCFL)

Theorem: DCFLs are closed under complement
(complement of \(L\) in \(\Sigma^*\) is \((\Sigma^* - L)\))

Proof attempt:
- swap accept/reject states
- problem: might enter infinite loop before reading entire string
- machine for complement must accept in these cases, and read to end of string
Example of problem

\[ \begin{align*}
&0, \varepsilon \rightarrow \varepsilon \\
&1, \varepsilon \rightarrow \varepsilon \\
&\varepsilon, \varepsilon \rightarrow \varepsilon
\end{align*} \]

Language of this DPDA is $\{ \varepsilon \}$

Deterministic PDA

Proof:
- convert machine into "normal form"
  - always reads to end of input
  - always enters either an accept state or single distinguished "reject" state
- step 1: keep track of when we have read to end of input
- step 2: eliminate infinite loops

Deterministic PDA

Step 1: keep track of when we have read to end of input

Step 2: eliminate infinite loops
Deterministic PDA

step 2: eliminate infinite loops
– on input x, if infinite loop, then:

\[ \text{stack height} \]

\[ \text{time} \]

\[ i_0 \]

\[ i_1 \]

\[ i_2 \]

\[ i_3 \]

infinite sequence \( i_0 < i_1 < \ldots \) such that for all \( i_k \), stack height never decreases below \( \text{ht}(i_k) \) after time \( i_k \).

Deterministic PDA

– infinite seq. \( i_0 < i_1 < \ldots \) such that for all \( k \), stack height never decreases below \( \text{ht}(i_k) \) after time \( i_k \).

– infinite subsequence \( j_0 < j_1 < \ldots \) such that same transition is applied at each time \( j_k \).

– never see any stack symbol below \( t \) from \( j_k \) on.

– we are in a periodic, deterministic sequence of stack operations independent of the input.

Deterministic PDA

– finishing up…
– have a machine \( M \) with no infinite loops
– therefore it always reads to end of input
– either enters an accept state \( q' \), or enters “reject” state \( r' \)
– now, can swap: make \( r' \) unique accept state to get a machine recognizing complement of \( L \)

Summary

• Nondeterministic Pushdown Automata (NPDA)
• Context-Free Grammars (CFGs) describe Context-Free Languages (CFLs)
  – terminals, non-terminals
  – productions
  – yields, derivations
  – parse trees

• NDPAs and CFGs are equivalent
• CFL Pumping Lemma is used to show certain languages are not CFLs
Summary

- Deterministic PDAs recognize DCFLs
- DCFLs are closed under complement
- There is an efficient algorithm (based on dynamic programming) to determine if a string $x$ is generated by a given grammar $G$.

So far...

- Several models of computation
  - Finite automata
  - Pushdown automata
- Fail to capture our intuitive notion of what is computable
- Regular languages
- Context-free languages
- All languages

So far...

- We proved (using constructions of FA and NPDAs and the two pumping lemmas):
  - Regular languages
  - Context-free languages
  - All languages
    - $\{w : w \in \{a,b\}^* \text{ has an even # of } b\}$
    - $\{a^n b^n : n \geq 0 \}$
    - $\{a^n b^n c^n : n \geq 0 \}$

A more powerful machine

- Limitation of NPDA related to fact that their memory is stack-based (last in, first out)
- What is the simplest alteration that adds general-purpose "memory" to our machine?
- Should be able to recognize, e.g., $\{a^n b^n : n \geq 0 \}$

Turing Machines

- Informal description:
  - Input written on left-most squares of tape
  - Rest of squares are blank
  - At each point, take a step determined by
    - Current symbol being read
    - Current state of finite control
  - A step consists of:
    - Writing new symbol
    - Moving read/write head left or right
    - Changing state

- New capabilities:
  - Infinite tape
  - Can read or write to tape
  - Read/write head can move left and right