CS21 Decidability and Tractability
Lecture 8
January 21, 2022

Outline
• equivalence of NPDAs and CFGs
• non context-free languages via CFL Pumping Lemma

NPDA, CFG equivalence

Theorem: a language $L$ is recognized by a NPDA iff $L$ is described by a CFG.

Must prove two directions:
$(\Rightarrow)$ $L$ is recognized by a NPDA implies $L$ is described by a CFG.
$(\Leftarrow)$ $L$ is described by a CFG implies $L$ is recognized by a NPDA.

Proof of $(\Rightarrow)$: $L$ is recognized by a NPDA implies $L$ is described by a CFG.

– harder direction
– first step: convert NPDA into “normal form”:
  • single accept state
  • empties stack before accepting
  • each transition either pushes or pops a symbol

NPDA, CFG equivalence

– main idea: non-terminal $A_{p,q}$ generates exactly the strings that take the NPDA from state $p$ (w/ empty stack) to state $q$ (w/ empty stack)

– then $A_{\text{start, accept}}$ generates all of the strings in the language recognized by the NPDA.

• Two possibilities to get from state $p$ to $q$: 

  \[
  \begin{align*}
  \text{generated by } A_{p,r} & \\
  \text{generated by } A_{r,q} & \\
  \end{align*}
  \]
NPDA, CFG equivalence

- NPDA $P = (Q, \Sigma, \Gamma, \delta, \text{start}, \{\text{accept}\})$
- CFG $G$:
  - non-terminals $V = \{A_{p,q} : p, q \in Q\}$
  - start variable $A_{\text{start}}$, accept
  - productions:
    for every $p, r, q \in Q$, add the rule
    $A_{p,r} \rightarrow A_{p,q}A_{r,s}$

NPDA, CFG equivalence

- Two possibilities to get from state $p$ to $q$:
  - stack height
  - input string taking NPDA from $p$ to $q$

NPDA, CFG equivalence

- NPDA $P = (Q, \Sigma, \Gamma, \delta, \text{start}, \{\text{accept}\})$
- CFG $G$:
  - non-terminals $V = \{A_{p,q} : p, q \in Q\}$
  - start variable $A_{\text{start}}$, accept
  - productions:
    for every $p, r, q \in Q$, $d \in \Gamma$ and $a, b \in (\Sigma \cup \{\epsilon\})$
    if $(r, d) \in \delta(p, a, \epsilon)$, and
    $(q, \epsilon) \in \delta(s, b, d)$, add the rule
    $A_{p,q} \rightarrow aA_{r,s}b$

NPDA, CFG equivalence

- two claims to verify correctness:
  1. if $A_{p,q}$ generates string $x$, then $x$ can take NPDA $P$ from state $p$ (w/ empty stack) to $q$ (w/ empty stack)
  2. if $x$ can take NPDA $P$ from state $p$ (w/ empty stack) to $q$ (w/ empty stack), then $A_{p,q}$ generates string $x$

NPDA, CFG equivalence

1. if $A_{p,q}$ generates string $x$, then $x$ can take NPDA $P$ from state $p$ (w/ empty stack) to $q$ (w/ empty stack)
   - induction on length of derivation of $x$
   - base case: 1 step derivation. must have only terminals on rhs. In $G$, must be production of form $A_{p,p} \rightarrow \epsilon$. 

January 21, 2022 CS21 Lecture 8
NPDA, CFG equivalence

1. If \( A_{p,q} \) generates string \( x \), then \( x \) can take NPDA \( P \) from state \( p \) (w/ empty stack) to \( q \) (w/ empty stack)
   - Assume true for derivations of length at most \( k \), prove for length \( k+1 \).
   - Verify case: \( A_{p,q} \rightarrow A_{p,r}A_{r,q} \rightarrow^{k} x = yz \)
   - Verify case: \( A_{p,q} \rightarrow aA_{r,b} \rightarrow^{k} x = ayb \)

NPDA, CFG equivalence

2. If \( x \) can take NPDA \( P \) from state \( p \) (w/ empty stack) to \( q \) (w/ empty stack), then \( A_{p,q} \) generates string \( x \)
   - Induction step. Assume true for computations of length at most \( k \), prove for length \( k+1 \).
   - If stack becomes empty sometime in the middle of the computation (at state \( r \))
     • \( y \) is read going from state \( p \) to \( r \) \((A_{p,r} \rightarrow^{*} y)\)
     • \( z \) is read going from state \( r \) to \( q \) \((A_{r,q} \rightarrow^{*} z)\)
     • Conclude: \( A_{p,q} \rightarrow A_{p,r}A_{r,q} \rightarrow^{*} yz = x \)

NPDA, CFG equivalence

2. If \( x \) can take NPDA \( P \) from state \( p \) (w/ empty stack) to \( q \) (w/ empty stack), then \( A_{p,q} \) generates string \( x \)
   - If stack becomes empty only at beginning and end of computation.
     • First step: state \( p \) to \( r \), read \( a \), push \( d \)
     • Go from state \( r \) to \( s \), read string \( y \) \((A_{r,s} \rightarrow^{*} y)\)
     • Last step: state \( s \) to \( q \), read \( b \), pop \( d \)
     • Conclude: \( A_{p,q} \rightarrow aA_{r,s}b \rightarrow^{*} ayb = x \)

Pumping Lemma for CFLs

CFL Pumping Lemma: Let \( L \) be a CFL.
There exists an integer \( p \) ("pumping length") for which every \( w \in L \) with \( |w| \geq p \) can be written as
\[ w = uvxyz \]
such that
1. For every \( i \geq 0 \), \( uv^{i}xy^{i}z \in L \), and
2. \( |vy| > 0 \), and
3. \( |vxy| \leq p \).
CFL Pumping Lemma Example

Theorem: the following language is not context-free:
\[ L = \{ a^n b^n c^n : n \geq 0 \} \]

Proof:
- let \( p \) be the pumping length for \( L \)
- choose \( w = a^p b^p c^p \)
- \( w = \text{aaa...abbb...bccc...c} \)
- \( w = uvxyz \), with \( |vy| > 0 \) and \( |vxy| \leq p \).

CFL Pumping Lemma Example

- possibilities:
  \( w = \text{aaa...abbb...bccc...c} \)
  (if \( v \) or \( y \) contain more than one type of symbol, then pumping on them might produce a string with equal numbers of a's, b's, and c's. But they will be out of order.)

CFL Pumping Lemma Example

- possibilities:
  \( w = \text{aaa...abbb...bccc...c} \)
  (if \( v \) or \( y \) contain more than one type of symbol, then pumping on them might produce a string with equal numbers of a's, b's, and c's. But they will be out of order.)

CFL Pumping Lemma Example

\[ L = \{ xx : x \in \{0,1\}^* \} \]

- try \( w = 0^p 1^p 0^p 1^p \)
- \( w = uvxyz \), with \( |vy| > 0 \) and \( |vxy| \leq p \).
- case: \( vxy \) in first half.
  - then \( uv^2 xy^2 z = {}^0?...?1?...? \)
- case: \( vxy \) in second half.
  - then \( uv^2 xy^2 z = {}^0?...?0?...?1 \)
- case: \( vxy \) straddles midpoint
  - then \( uv^2 xy^2 z = uxz = 0^p 10^p 1^p \) with \( i \neq 2p \) or \( j \neq 2p \)

CFL Pumping Lemma Example

Theorem: the following language is not context-free:
\[ L = \{ xx : x \in \{0,1\}^* \} \]

Proof:
- let \( p \) be the pumping length for \( L \)
- try \( w = 0^p 1^p 0^p 1^p \)
- can this be pumped?

CFL Pumping Lemma Example

- possibilities:
  \( w = \text{aaa...abbb...bccc...c} \)
  (if \( v \) or \( y \) each contain only one type of symbol, then pumping on them produces a string not in the language)

CFL Pumping Lemma for CFLs

Pumping Lemma for CFLs: Let \( L \) be a CFL. There exists an integer \( p \) ("pumping length") for which every \( w \in L \) with \( |w| \geq p \) can be written as
\[ w = uvxyz \quad \text{such that} \]
1. for every \( i \geq 0 \), \( uv^i xy^i z \in L \), and
2. \( |vy| > 0 \), and
3. \( |vxy| \leq p \).
CFL Pumping Lemma

**Proof:** consider a parse tree for a very long string \( w \in L \):

```
S
A
<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
</table>
A
  |   |
D b
  |   |
A
  |   |
B b
  |   |
A
  |   |
C b
  |   |
A
  |   |
D b
  |   |
S
```

some non-terminal must repeat on long path

- how large should pumping length \( p \) be?
- need to ensure other conditions:
  - \(|vy| > 0\)
  - \(|vxy| \leq p\)
- \( b = \max \) # symbols on rhs of any production (assume \( b \geq 2 \))
- if parse tree has height \( \leq h \), then string generated has length \( \leq b^h \) (so length > \( b^h \) implies height > \( h \))

- let \( m \) be the # of nonterminals in the grammar
- to ensure path of length at least \( m+2 \), require
  - \(|w| \geq p = b^{m+2}\)
- since \(|w| > b^{m+1}\), any parse tree for \( w \) has height \( \geq m+1 \)
- let \( T \) be the smallest parse tree for \( w \)
- longest root-leaf path must consist of \( \geq m+1 \) non-terminals and 1 terminal.

- must be a repeated non-terminal \( A \) on long path
- select a repetition among the lowest \( m+1 \) non-terminals on path.
- pictures show that for every \( i \geq 0 \), \( uv^i xy^i z \in L \)
- is \(|vy| > 0 \)?
  - smallest parse tree \( T \) ensures
  - is \(|vxy| \leq p \)?
    - red path has length \( \leq m+2 \), so \( \leq b^{m+2} = p \) leaves
### Chomsky Normal Form

- Useful to deal only with CFGs in a simple normal form
- Most common: Chomsky Normal Form (CNF)
- Definition: every production has form
  \[ A \rightarrow BC \quad \text{or} \quad S \rightarrow \varepsilon \quad \text{or} \quad A \rightarrow a \]
  where \( A, B, C \) are any non-terminals (and \( B, C \) are not \( S \)) and \( a \) is any terminal.

### Theorem
Every CFL is generated by a CFG in Chomsky Normal Form.

### Proof
Exercise or in book…