CS21
Decidability and Tractability
Lecture 8
January 22, 2021

Outline

• equivalence of NPDAs and CFGs (continued)
• non context-free languages via CFL Pumping Lemma
• Chomsky Normal Form and deciding CFLs

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.

NPDA, CFG equivalence

**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.
NPDA, CFG equivalence

- NPDA P = (Q, Σ, Γ, δ, start, {accept})
- CFG G:
  - non-terminals V = {A_{p,q} : p, q ∈ Q}
  - start variable A_{start, accept}
  - productions:
    for every p, r, q ∈ Q, add the rule
    $$A_{p,q} \rightarrow A_{p,r}A_{r,q}$$

NPDA, CFG equivalence

- NPDA P = (Q, Σ, Γ, δ, start, {accept})
- CFG G:
  - non-terminals V = {A_{p,q} : p, q ∈ Q}
  - start variable A_{start, accept}
  - productions:
    for every p ∈ Q, add the rule
    $$A_{p,p} \rightarrow \epsilon$$

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

- Two possibilities to get from state p to q:
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: $A_{p,q} \rightarrow A_p A_{r,q} \rightarrow^k x = \epsilon$.
   - Verify case: $A_{p,q} \rightarrow a A_{r,s} b \rightarrow^k x = ayb$.

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: $A_{p,q} \rightarrow a A_{r,s} b \rightarrow^* ayb = x$.

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 $i \geq 0$, $uv^i x y^i z \in L$, 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 \( w = a^p b^p c^p \)
  - \( w = aaaa...abbb...bccc...c \)
  - \( w = uvxyz \), with \(|y| > 0 \) and \(|xy| \leq p \).

---

CFL Pumping Lemma Example

- **possibilities**:
  \( w = aaaa...abbb...bccc...c \)

(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 – if \( vy \) contains equal numbers of a's, b's, and c's. But they will be out of order.)

---

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 \( w = 0^p10^p10^p \)
  - can this be pumped?

---

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

\[ w = uvxy \]

such that

1. for every \( i \geq 0 \), \( uv^i x y^i \in L \), and
2. \(|xy| > 0 \), and
3. \(|vxy| \leq p \).
CFL Pumping Lemma

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

```
S → A
  ∣
  v
A → A
  ∣
  v
S → A
  ∣
  v
A → A
  ∣
  v
...
```

some non-terminal must repeat on long path

January 22, 2021  CS21 Lecture 8  25

---

CFL Pumping Lemma

• Schematic proof:

```
S
  ∣
  △
A
```

January 22, 2021  CS21 Lecture 8  26

---

CFL Pumping Lemma

– how large should pumping length \( p \) be?
– need to ensure other conditions:

\[
|vy| > 0 \quad |vxy| \leq p
\]

\[b = \max \text{ # symbols on rhs of any production (assume } b \geq 2)\]

– if parse tree has height \( \leq h \), then string generated has length \( \leq b^h \) (so length \( > b^h \) implies height \( > h \))

January 22, 2021  CS21 Lecture 8  28

---

CFL Pumping Lemma

– 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 \( > m+1 \)
– let \( T \) be the smallest parse tree for \( w \)
– longest root-leaf path must consist of \( \geq m+1 \) non-terminals and 1 terminal.

January 22, 2021  CS21 Lecture 8  29

---

CFL Pumping Lemma

– 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 \( i \geq 0, uv^iwxz \in L \)
– 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

January 22, 2021  CS21 Lecture 8  30