NDPAs recognize CFLs.

What is the intuitive relationship?
Forget about PDAs: Let’s write code to interpret a CFG.

#### CFGs

- $V$: set of variables
- $\Sigma$: set of terminals
- $R$: set of rules
- $S \in V$: start variable
- $V \cap \Sigma = \emptyset$

1. Start at start variable.
2. Substitute variables.
3. Match against input.

We only have a stack.
How do we store strings on an NPDA’s stack?

$L = \{ w \mid w \text{ has the same number of 0s and 1s} \}$

Grammar $G$:

\[
A \rightarrow 0A1A \mid 1A0A \\
A \rightarrow \varepsilon
\]
Can CFGs generate any language an NPDA recognizes?

Plan:

Given an NPDA $P$, construct a restricted NPDA that is equivalent, then construct a CFG $G$.

$P = (Q, \Sigma, \Gamma, \delta, q_0, \{q_{\text{accept}}\})

P = (Q, \Sigma, \Gamma, \delta, q_0, \{q_{\text{accept}}\})

\downarrow

G = (V, \Sigma, R, S)$

Properties:
1. Single accept state.
2. Empties stack before accepting.
3. Each transition only pushes or pops.
\[ G = (V, \Sigma, R, S) \]
\[ V = \{ A_{pq} \mid p, q \in Q \} \]
\[ S = A_{q_0, q_{\text{accept}}} \]

For each \( p, q, r, s \in Q, t \in \Gamma, a, b \in \Sigma : \)
if \( \delta(p, a, \varepsilon) \) contains \( (r, t) \) and
\( \delta(s, b, t) \) contains \( (q, \varepsilon) \),
add rule \( A_{pq} \rightarrow aA_{rs}b \).

If we have an NPDA \( P \) that goes from \( p \) to \( q \) on input \( x \), does the rule \( A_{pq} \) in grammar \( G \) generate \( x \)?
Basis: In 0 steps, P reads the empty string. G generates the empty string with $A_{pp}$.

Induction: Assume true for $k$ steps. Suppose NPDA P goes from $p$ to $q$ on input $x$ in $k+1$ steps.

Either the stack is empty at both ends of the computation, or it is empty elsewhere.

1. Suppose $t$ is pushed first, $a$ in read first, and $b$ is read last and $y$ is read in between. Then $A_{pq}$ generates this input, and the rest occurs in $k-1$ steps.
2. Suppose the stack becomes empty at state \( r \) between \( p \) and \( q \). \( y \) is read between \( p \) and \( r \), \( z \) is read between \( r \) and \( q \). \( A_{pq} \) generates this input, and \( A_{pr} \) and \( A_{rq} \) occur in \( k \) steps.

Some languages weren’t regular.

Is the following language regular?

\[
L = \{ww^R\} \\
\Sigma = \{0, 1\}
\]

Pumping Lemma

For every regular language \( L \) there exists some integer \( p \) where for every string \( s \) in \( L \) of length at least \( p \), \( s = xyz \) and \( y \) can be repeated, \( |y| > 0 \), and \( |xy| \leq p \).

\[
L = \{ww^R\} 
\]
Is the following language regular?

\( L = \{ww^R\} \)

1. Identify some property P that is true for all regular languages.

There exists some DFA that recognizes any regular language and accepts some string s of length p for which we can apply the pumping lemma.

2. Assume P holds true for language L.

Assume L is regular and thus has a DFA and is pumpable.

3. Obtain a contradiction, thereby showing L is not regular.

\( s = 0^p110^p \)

\(|xy| \leq p, |y| > 0 \Rightarrow y \in \{0^+\} \Rightarrow xy^0z \notin L\)

### CFL Pumping Lemma

For every CFL language L there exists some integer p where for every string s in L of length at least p, s = uvxyz and v,y can be repeated, |vy| > 0, and |vxy| \( \leq p \).

Is the following language in CFL?

\( L = \{ww\} \)

\( \Sigma = \{0, 1\} \)

Is the following language in CFL?

\( L = \{ww\} \)

NO

\( s = 0^p1^p0^p1^p \)
Is the following language in **CFL**?

\[ L = \{0^n1^n2^n \mid n \geq 0\} \]

\[ \Sigma = \{0, 1, 2\} \]

**NO**

\[ s = 0^p1^p2^p \]

Reading: Sipser 2.2, 2.3