

Computational Models - Lecture 4¹

Handout Mode

Iftach Haitner and Yishay Mansour.

Tel Aviv University.

April 3/8, 2013

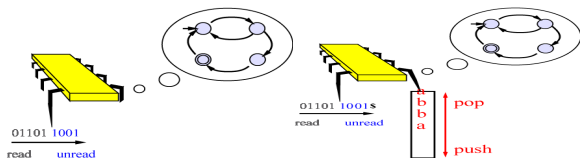
¹Based on frames by Benny Chor, Tel Aviv University, modifying frames by Maurice Herlihy, Brown University.

Talk Outline

- **Context Free** Grammars/Languages (CFG/CFL)
- Algorithmic issues for CFL
- **Pumping Lemma** for context free languages

Next week:

- Push Down Automata (**PDA**)
- **Equivalence** of CFGs and PDAs



- Sipser's book, 2.1, 2.2 & 2.3

Short Overview of the Course

So far we saw

- finite automata,
- regular languages,
- regular expressions,
- Myhill-Nerode theorem
- pumping lemma for **regular languages**.

We now introduce stronger machines and languages with more expressive power:

- pushdown automata,
- context-free languages,
- context-free grammars,
- pumping lemma for context-free languages.

Context Free Grammars (CFG)

An example of a context free grammar, G_1 :

- $A \rightarrow 0A1$
- $A \rightarrow B$
- $B \rightarrow \#$

Terminology:

- Each line is a **substitution rule** or **production**.
- Each rule has the form: **symbol** \rightarrow **string**.
The left-hand symbol is a **variable** (usually upper-case).
- A **string** consists of **variables** and **terminals**.
- One variable is the **start variable** (lhs of top rule).

Rules for Generating Strings

- Write down the start variable.
- Pick a variable written down in current string and a derivation that starts with that variable.
- Replace that variable with right-hand side of that derivation.
- Repeat until no variables remain.
- Return final string (concatenation of terminals).

Process is inherently **non deterministic**.

Example

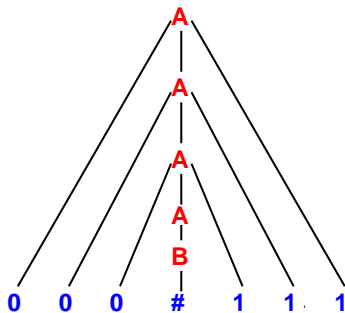
Grammar G_1 :

- $A \rightarrow 0A1$
- $A \rightarrow B$
- $B \rightarrow \#$

Derivation with G_1 :

$A \rightarrow 0A1$
 $\rightarrow 00A11$
 $\rightarrow 000A111$
 $\rightarrow 000B111$
 $\rightarrow \mathbf{000\#111}$

Parsing Trees



Indifferent to derivation **order**.

Question 1

What strings can be generated in this way from the grammar G_1 ?

Answer: Exactly those of the form $0^n\#1^n$ ($n \geq 0$).

Context-Free Languages (CFL)

The language generated in this way is called the **language of the grammar**.

For example, $\mathcal{L}(G_1) = \{0^n \# 1^n : n \geq 0\}$.

Any language generated by a context-free grammar is called a **context-free language**.

A Useful Abbreviation

Rules with same variable on left hand side

$$A \rightarrow 0A1$$

$$A \rightarrow B$$

are written as:

$$A \rightarrow 0A1 \mid B$$

English-like Sentences

A grammar G_2 to describe a few English sentences:

$\langle \text{SENTENCE} \rangle \rightarrow \mathcal{NP} \langle \text{VERB} \rangle$
 $\mathcal{NP} \rightarrow \langle \text{ARTICLE} \rangle \langle \text{NOUN} \rangle$
 $\langle \text{NOUN} \rangle \rightarrow \text{boy} \mid \text{girl} \mid \text{flower}$
 $\langle \text{ARTICLE} \rangle \rightarrow \text{a} \mid \text{the}$
 $\langle \text{VERB} \rangle \rightarrow \text{touches} \mid \text{likes} \mid \text{sees}$

Deriving English-like Sentences

A specific derivation in G_2 :

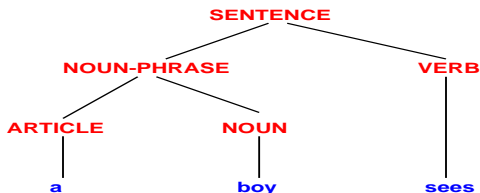
< SENTENCE > \rightarrow \mathcal{NP} < VERB >
 \rightarrow < ARTICLE >< NOUN >< VERB >
 \rightarrow a < NOUN >< VERB >
 \rightarrow a boy < VERB >
 \rightarrow a boy sees

More strings generated by G_2 :

a flower sees
the girl touches

Derivation and Parse Tree

- < SENTENCE > \rightarrow \mathcal{NP} < VERB >
- \rightarrow < ARTICLE >< NOUN >< VERB >
- \rightarrow a < NOUN >< VERB >
- \rightarrow a boy < VERB >
- \rightarrow a boy sees



Formal Definition

A **context-free grammar** is a 4-tuple (V, Σ, R, S) , where

- V is a finite set of **variables**
- Σ is a finite set of **terminals**
- R is a finite set of **rules**: each rule is a variable and a finite string of variables and terminals.
- S is the **start symbol**.
- If u and v are strings of **variables** and **terminals**, and $A \rightarrow w$ is a **rule** of the grammar, then uAv **yields** uwv , written $uAv \rightarrow uwv$.
- $u \xrightarrow{*} v$ if $u = v$, or $u \rightarrow u_1 \rightarrow \dots \rightarrow u_k \rightarrow v$ for some sequence u_1, u_2, \dots, u_k

Definition 2

The **language of the grammar** G , denoted $\mathcal{L}(G)$, is $\{w \in \Sigma^* : S \xrightarrow{*} w\}$

where $\xrightarrow{*}$ is determined by G .

Example 1

$G_4 = (\{S\}, \{a, b\}, R, S)$.

R (Rules): $S \rightarrow aSb \mid SS \mid \epsilon$

Some words in the language: *aabb*, *aababb*.

Question 3

What **is** this language?

Hint: Think of parentheses.

Example 2

$$G_5 = (\{S\}, \{a, b\}, R, S).$$

$$R \text{ (Rules): } S \rightarrow aSa \mid bSb \mid \epsilon$$

Some words in the language: *abba, aabaabaa*.

Question 4

What is this language?

$$\mathcal{L}(G_5) = \{ww^R : w \in \{a, b\}^*\}$$

Example 3

$$G_6 = (\{S, A, B\}, \{a, b\}, R, S)$$

R (Rules):

$$S \rightarrow aB \mid bA \mid \varepsilon$$

$$A \rightarrow a \mid aS \mid bAA$$

$$B \rightarrow b \mid bS \mid aBB$$

Some words in the language: *aababb*, *baabba*.

Question 5

What is this language?

$$\mathcal{L}(G_6) = \{w \in \{a, b\}^* : \#_a(w) = \#_b(w)\}$$

$\#_x(w)$ — number of occurrences of x in w

Proving $\mathcal{L}(G_6) = \{w \in \{a, b\}^* : \#_a(w) = \#_b(w)\}$

What do we need to show?

- 1 If w generated by G_6 then $\#_a(w) = \#_b(w)$.
- 2 If $\#_a(w) = \#_b(w)$ then w is generated by G_6 .

Claim 6

If $S \xrightarrow{*} w$, then $\#_a(w) + \#_A(w) = \#_b(w) + \#_B(w)$.

Claim 7

For $w \in \{a, b\}^*$ let $k(w) = \#_a(w) - \#_b(w)$. Then:

- 1 If $k(w) = 0$, then $S \xrightarrow{*} wS$
- 2 If $k(w) > 0$, then $S \xrightarrow{*} wB^k$
- 3 If $k(w) < 0$, then $S \xrightarrow{*} wA^k$

Are we done?!

How we prove these claims?

Proving Claim 7

Proof by induction on $|w|$:

- Basis: $w = \epsilon$ then $k(w) = 0$. Since $S \xrightarrow{*} S$, then $S \xrightarrow{*} \epsilon S$
- Induction step: for $w \in \{a, b\}^n$ write $w = w'\sigma$ with $|w'| = n - 1$

Suppose $k' = k(w') = (\#_a(w') - \#_b(w')) > 0$ and $\sigma = a$

By the induction hypothesis $S \xrightarrow{*} w'B^{k'}$

Since $B \rightarrow aBB$, then $S \xrightarrow{*} wB^{k'+1} = wB^{k(w)}$

To complete the proof, need to show for $\sigma = b$ and for $k' \leq 0$

Designing Context-Free Grammars

No recipe in general, but few rules-of-thumb

- If CFG is the **union** of several CFGs, rename variables (**not terminals**) so they are disjoint, and add new rule $S \rightarrow S_1 \mid S_2 \mid \dots \mid S_i$.
- For languages (like $\{0^n \# 1^n : n \geq 0\}$), with **linked** substrings, a rule of form $R \rightarrow uRv$ is helpful to force desired relation between substrings.
- For a regular language, grammar “follows” a DFA for the language (see next frame).

CFG for Regular Languages

Given a DFA : $M = (Q, \Sigma, \delta, q_0, F)$

CFG G for $\mathcal{L}(M)$:

- 1 Let R_0 be the starting variable
- 2 Add rule $R_i \rightarrow aR_j$, for any $q_i, q_j \in Q$ and $a \in \Sigma$ with $\delta(q_i, a) = q_j$
- 3 Add rule $R_i \rightarrow \varepsilon$, for any $q_i \in F$

Claim 8

$$\mathcal{L}(G) = \mathcal{L}(M)$$

Proof?

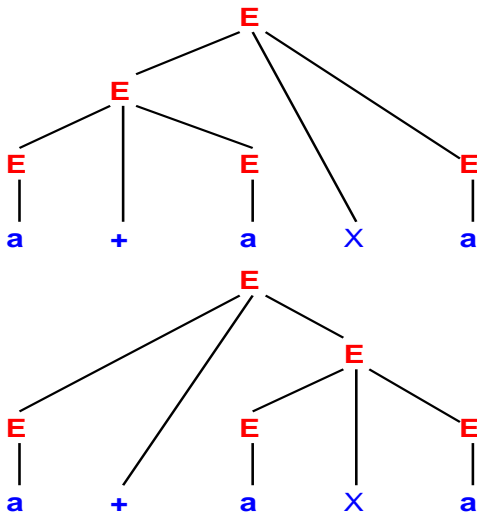
Claim 9

$$R_0 \xrightarrow{*} wR_j \text{ iff } M(w) = q_j.$$

Proof? Next class we'll see alternative proof via "Push-Down Automata"

Ambiguity in CFLs

Grammar G : $E \rightarrow E + E \mid E \times E \mid (E) \mid a$



Arithmetic Example

Consider the grammar $G' = (V, \Sigma, R, E)$, where

- $V = \{E, T, F\}$
- $\Sigma = \{a, +, \times, (,)\}$
- Rules:
 - $E \rightarrow E + T \mid T$
 - $T \rightarrow T \times F \mid F$
 - $F \rightarrow (E) \mid a$

Claim 10

$$\mathcal{L}(G') = \mathcal{L}(G)$$

Proof?, but G' is not ambiguous.

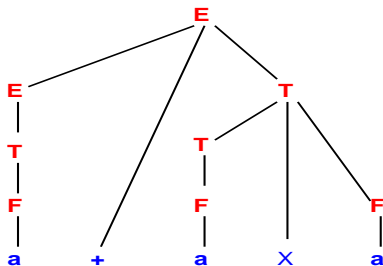
Parsing Tree of G' for $a + a \times a$

G' :

$$E \rightarrow E + T \mid T$$

$$T \rightarrow T \times F \mid F$$

$$F \rightarrow (E) \mid a$$



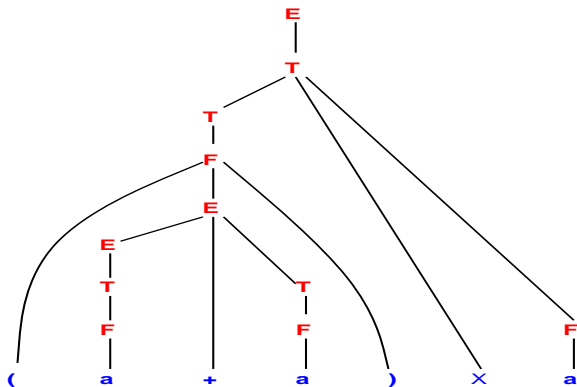
Parsing Tree of G' for $(a + a) \times a$

G' :

$$E \rightarrow E + T \mid T$$

$$T \rightarrow T \times F \mid F$$

$$F \rightarrow (E) \mid a$$



Ambiguity

Definition 11

A string w is derived **ambiguously** from grammar G , if w has two or more **different** parse trees that generate it from G . A CFG is **ambiguous**, if it ambiguously derives a string.

Ambiguity is usually not only a syntactic notion but also **semantic**, implying multiple meanings for the same string. Think of $a + a \times a$ from last grammar.

It is **sometime** possible to **eliminate** ambiguity by finding a different context free grammar generating the same language. This is true for the **arithmetic expressions** grammar.

Some languages are **inherently** ambiguous.

Example: $\{1^i 2^j 3^k : i = j \vee j = k\}$ Proof?

Part I

Checking Membership

Checking Membership in a CFL

Challenge

Given a CFG G and a string w , decide whether $w \in \mathcal{L}(G)$?

Initial Idea: Design an algorithm that tries **all derivations**.

Problem: If G does **not** generate w , we'll never stop.

Chomsky Normal Form (CNF)

A **simplified**, canonical form of context free grammars.

$G = (V, \Sigma, R, S)$ is in a CNF, if every rule in R has one of the following forms:

$$\begin{aligned} A &\rightarrow a, & A \in V \wedge a \in \Sigma \\ A &\rightarrow BC, & A \in V \wedge B, C \in V \setminus \{S\} \\ S &\rightarrow \varepsilon. \end{aligned}$$

Simpler to analyze: each derivation adds (at most) a single terminal, S only appears once, ε appears only at the empty word

Theorem 12

Any context-free language is generated by a context-free grammar in Chomsky Normal Form.

Proof Idea:

- Add new start symbol S_0 .
- Eliminate all ϵ rules of the form $A \rightarrow \epsilon$.
- Eliminate all “unit” rules of the form $A \rightarrow B$.
- Patch up rules so that grammar generates the same language.
- Convert remaining “long rules” to proper form.

CNF: Example

$$S \rightarrow ASA \mid aB$$

$$A \rightarrow B \mid S$$

$$B \rightarrow b \mid \varepsilon$$

Is transformed into:

$$S_0 \rightarrow AA_1 \mid UB \mid a \mid SA \mid AS$$

$$S \rightarrow AA_1 \mid UB \mid a \mid SA \mid AS$$

$$A \rightarrow b \mid AA_1 \mid UB \mid a \mid SA \mid AS$$

$$A_1 \rightarrow SA$$

$$U \rightarrow a$$

$$B \rightarrow b$$

CNF: Bounded Derivation Length

Lemma 13

For a CNF grammar G and $w \in \mathcal{L}(G)$ with $|w| = n \geq 1$, it holds that w has a derivation of length $2n - 1$.

Proof? consider the parsing tree for w

Advantage: Easier to check whether $w \in \mathcal{L}(G)$

Checking Membership for Grammars in CNF Form

Given a CNF grammar $G = (V, \Sigma, R, S)$, we build a function

$\text{Derive}(A, x)$ that returns **TRUE** iff $A \xrightarrow{*} x$.

Algorithm 14 ($\text{Derive}(A, x)$)

- If $x = \varepsilon$: if $A \rightarrow \varepsilon \in R$ (i.e., $A = S$) return **TRUE**, otherwise return **FALSE**.
- If $|x| = 1$: if $A \rightarrow x \in R$ return **TRUE**, otherwise return **FALSE**.
- For each $A \rightarrow BC$ and each partition $x = x_1 x_2$:
 - ▶ Call $\text{Derive}(B, x_1)$ and $\text{Derive}(C, x_2)$.
 - ▶ Return **TRUE** if *both* return **TRUE**.
- Return **FALSE**.

Test whether $w \in \mathcal{L}(G)$ by calling $\text{Derive}(S, w)$

Correctness?

- Procedure **Derive** can also output a **parse tree** for w
- Have we **critically** used that G is in CNF?

Time Complexity of Derive

What is the time complexity $T: \mathbb{N} \mapsto \mathbb{N}$ of **Derive**?

- ▶ Each recursive call tests $|R|$ rules and n partitions.

$$T(n) \leq |R| \cdot n \cdot 2T(n-1)$$

$$T(n) \in O((|R| \cdot n)^n).$$

- ▶ Still exponential...

Efficient Algorithm

- Keep in memory the results of $\text{Derive}(A, x)$.
 - ▶ Number of different inputs: $|V| \cdot n^2$.
 - ▶ Only $|V| \cdot n^2$ calls, each takes $O(|R| \cdot n)$.
 - ▶ $T(n) \in O(|R| \cdot n^3 \cdot |V|)$.
- Polynomial time!
- This approach is called **Dynamic Programming**
Basic idea:
 - ▶ If number of different inputs is limited, say $I(n)$.
 - ▶ Each run (excluding recursive calls) takes at most $R(n)$ time
 - ▶ Total running time is bounded by $T(n) \leq R(n)I(n)$.

Part II

Non-Context-Free Languages

Proving a Language is **not** a CFL

- The **pumping lemma** for finite automata and **Myhill-Nerode** theorem are our tools for showing that languages are **not regular**.
- We will now show a similar **pumping lemma** for context-free languages.
- It is slightly more complicated . . .

Pumping Lemma for CFL (also known as, the $uvxyz$ Theorem)

Theorem 15

For any CFL \mathcal{L} there exists $\ell \in \mathbb{N}$ ("critical length"), such that for any $w \in \mathcal{L}$ with $|w| \geq \ell$, there exist $u, v, x, y, z \in \Sigma^*$ such that $w = uvxyz$ and

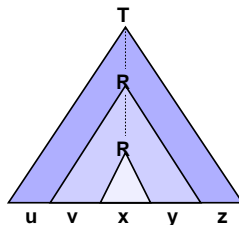
- For every $i \geq 0$: $uv^i xy^i z \in \mathcal{L}$
- $|vy| > 0$, ("non-triviality")
- $|vxy| \leq \ell$

Basic Intuition:

Let \mathcal{L} be a CFL and let w be a "very long" string in \mathcal{L} . Then w must have a "tall" parse tree.

Hence, some root-to-leaf path must repeat a symbol. **Ehy is that so?**

Basic Intuition cont.



We have: $T \xrightarrow{*} uRz$, $R \xrightarrow{*} vRy$, and $R \xrightarrow{*} x$.

Proof of Thm 15

Let G be a CFG and let $\mathcal{L} = \mathcal{L}(G)$.

- Let b be the max number of symbols in right-hand-side of any rule (what is b for a CNF grammar?).

Since no node in a parse tree of G has more than b children, at depth d such tree has at most b^d leaves.

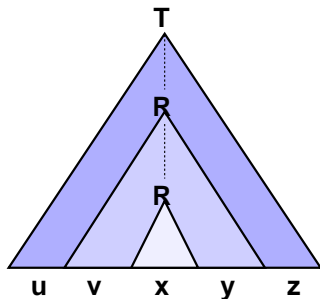
- Let $|V|$ be the number of variables in G , and set $\ell = b^{|V|+2}$.

Let w be a string with $|w| \geq \ell$, and let T be parse tree for w (with respect to G) with **fewest** nodes

- T has height $\geq |V| + 2$
- Some path in T has length $\geq |V| + 2$
- Such path **repeats** a variable R

Proof of Thm 15 cont.

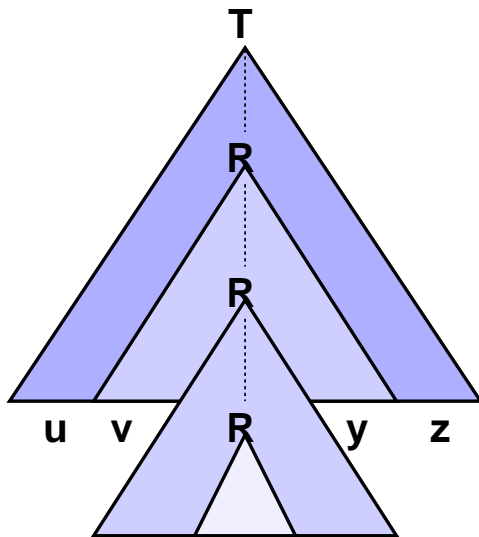
Set $w = uvxyz$



- Each occurrence of R produces a string
- Upper produces string vxy
- Lower produces string x

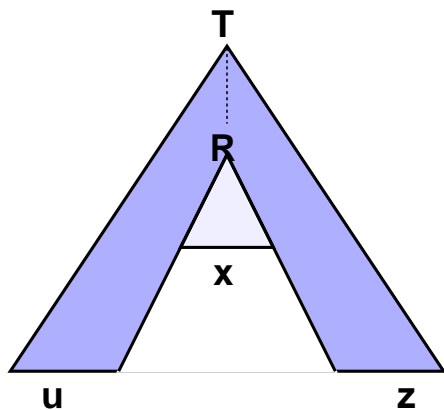
Proving $uv^ixy^iz \in \mathcal{L}$ for all $i > 1$

Replacing smaller by larger yields uv^ixy^iz , for $i > 0$.



Proving $uv^i xy^i z \in \mathcal{L}$ for $i = 0$

Replacing larger by smaller yields uxz .

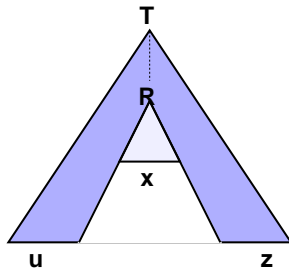


Together, they establish:

- $uv^i xy^i z \in \mathcal{L}$ for all $i \geq 0$

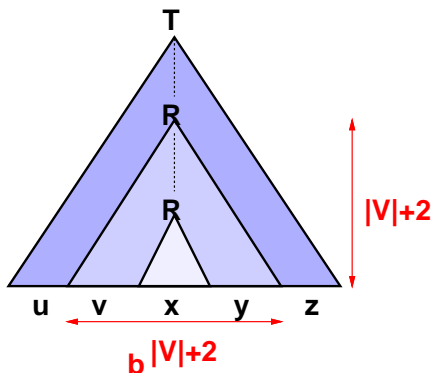
Proving $|vy| > 0$

If v and y are both ε , then



is a parse tree for w with **fewer nodes** than T , a contradiction.

Proving $|vxy| \leq \ell$



- Without loss of generality both occurrences of R lie in bottom $|V| + 1$ variables on the path.
- The upper occurrence of R (from now on R^1) generates vxy .
- Subtree rooted at R^1 is of height at most $|V| + 2$.
Hence, $|vxy| \leq b^{|V|+2} = \ell$.

Non CFL Example (1)

Claim 16

$\mathcal{L}_1 = \{a^n b^n c^n : n \in \mathbb{N}\}$ is not a CFL.

Proof: By contradiction. Assume \mathcal{L}_1 is a CFL with grammar G , let ℓ be the critical length of G and consider $w = a^\ell b^\ell c^\ell$. Let u, v, x, y, z be the strings with $w = uvxyz$ guaranteed by Thm 15 for w .

- Note that neither v nor y contain
 - ▶ both a 's and b 's, or
 - ▶ both b 's and c 's,(otherwise uv^2xy^2z would have out-of-order symbols).
- But if v and y contain only one letter, then uv^2xy^2z is imbalanced



Non CFL Example (2)

Claim 17

$\mathcal{L}_2 = \{a^i b^j c^k : 0 \leq i \leq j \leq k\}$ is not context free.

Proof: By contradiction. Assume \mathcal{L}_2 is a CFL with grammar G , let ℓ be the critical length of G and consider $w = a^\ell b^\ell c^\ell$. Let u, v, x, y, z be the strings with $w = uvxyz$ guaranteed by Thm 15 for w .

- Neither v nor y contains two distinct symbols, because otherwise uv^2xy^2z would have out-of-order symbols.
- vxy cannot be all the same letter (why?)
- $|vxy| \leq \ell$, so either
 - ▶ v contains only a 's and y contains only b 's, but then uv^2xy^2z has too few c 's.
 - ▶ v contains only b 's and y contains only c 's, but then uv^0xy^0z has too many a 's.



Non CFL Example (3)

Claim 18

$\mathcal{L}_3 = \{ww : w \in \{0, 1\}^*\}$ is not context-free.

Proof:

By contradiction. Assume \mathcal{L}_3 is a CFL with grammar G , let ℓ be the critical length of G and consider $w = 0^\ell 1^\ell 0^\ell 1^\ell$. Let u, v, x, y, z be the strings with $w = uvxyz$ guaranteed by **Thm 15** for w .

- Assuming vxy is in the first half of w , then uv^2xy^2z "moves" a 1 into the first position of second half.
- Assuming vxy is in the second half, then uv^2xy^2z "moves" a 0 into the last position of first half.
- Assuming vxy straddles the midpoint, then pumping *down* to uxz yields $0^\ell 1^i 0^j 1^\ell$ where i and j cannot both be ℓ .



Note that $\{ww^R : w \in \{0, 1\}^*\}$ is a CFL.