PROPERTIES OF CONTEXT-FREE LANGUAGES

a.y. 2023-2024

Non-terminals play variables

LEMMA
Let $\mathcal{G} = (V, T, S, \mathcal{P})$. Then every grammar $\mathcal{G}'$ obtained from $\mathcal{G}$ by renaming its non-terminals is such that $\mathcal{L}(\mathcal{G}) = \mathcal{L}(\mathcal{G}')$.

Non-terminals play variables
EXAMPLE

- For instance, any derivation for the grammar

$$\begin{array}{rcl} S & \to & aSbC \mid ab \\ C & \to & cC \mid c \end{array}$$

- Can be mimicked using the grammar

$$\begin{array}{rcl} R & \to & aRbD \mid ab \\ D & \to & cD \mid c \end{array}$$

- And viceversa

Cleaning up free grammars

LEMMA
Let $\mathcal{L}$ be a context-free language. Then there exists a context-free grammar $\mathcal{G}$ such that $\mathcal{L}(\mathcal{G}) = \mathcal{L} \setminus \{\epsilon\}$ and that has:

- No ϵ-production (i.e. no production of the shape $A \to \epsilon$)
- No unit production (i.e. no production of the shape $A \to B$)
- No useless non-terminal (i.e. non-terminals that never appear in some derivations of some strings of terminals)

Cleaning up free grammars
EXAMPLE

$$\begin{array}{lcl} S & \rightarrow & ABC \mid abc \\ A & \rightarrow & aB \mid \epsilon \\ B & \rightarrow & bA \mid C \\ C & \rightarrow & \epsilon \end{array}$$

Cleaning up free grammars
ϵ-PRODUCTION ELIMINATION

- Find **nullable** non-terminals, i.e., every non-terminal A such that $A \Rightarrow^* \epsilon$
  - **Base:** if $A \rightarrow \epsilon$ is a production, then A is nullable
  - **Iteration:** if $A \rightarrow Y_1 Y_2 \dots Y_n$ is a production and $Y_1, Y_2, \dots, Y_n$ are all nullable, then A is nullable
- Substitute each production $A \rightarrow Y_1 Y_2 \dots Y_n$ by a family of productions where combinations of nullable Y_is are removed from the body. (Exception: If all of Y_is are nullable do not take in the family the production $A \rightarrow \epsilon$)
- Eliminate every production $A \rightarrow \epsilon$

Cleaning up free grammars
EXAMPLE

$$
\begin{array}{lcl}
S & \rightarrow & ABC \mid abc \\
A & \rightarrow & aB \mid \epsilon \\
B & \rightarrow & bA \mid C \\
C & \rightarrow & \epsilon
\end{array}
$$

- Find **nullable** non-terminals
  - A and C nullable by $A \rightarrow \epsilon, C \rightarrow \epsilon$
  - B nullable by $B \rightarrow C$ and C nullable
  - S nullable by $S \rightarrow ABC$ and A, B, C nullable

FORMAL LANGUAGES AND COMPILERS Paola Quaglia, 2023

Cleaning up free grammars
EXAMPLE

$$
\begin{array}{lcl}
S & \rightarrow & ABC \mid abc \\
A & \rightarrow & aB \mid \epsilon \\
B & \rightarrow & bA \mid C \\
C & \rightarrow & \epsilon
\end{array}
$$

- Substitute each production $A \rightarrow Y_1 Y_2 \dots Y_n$ by a family of productions (if all of Y_is are nullable is useless to take in the family the production $A \rightarrow \epsilon$)

$$
\begin{array}{lcl}
S & \rightarrow & ABC \mid abc \mid AB \mid AC \mid BC \mid A \mid B \mid C \\
A & \rightarrow & aB \mid a \mid \epsilon \\
B & \rightarrow & bA \mid b \mid C \\
C & \rightarrow & \epsilon
\end{array}
$$

Cleaning up free grammars

EXAMPLE

$$\begin{array}{rcl} S & \to & ABC \mid abc \mid AB \mid AC \mid BC \mid A \mid B \mid C \\ A & \to & aB \mid a \mid \epsilon \\ B & \to & bA \mid b \mid C \\ C & \to & \epsilon \end{array}$$

- Eliminate every production $A \to \epsilon$

$$\begin{array}{rcl} S & \to & ABC \mid abc \mid AB \mid AC \mid BC \mid A \mid B \mid C \\ A & \to & aB \mid a \\ B & \to & bA \mid b \mid C \end{array}$$

Cleaning up free grammars

EXAMPLE

$$\begin{array}{rcl} S & \to & ABC \mid abc \mid AB \mid AC \mid BC \mid A \mid B \mid C \\ A & \to & aB \mid a \\ B & \to & bA \mid b \mid C \end{array}$$

- Useless non-terminal elimination

$$\begin{array}{rcl} S & \to & abc \mid AB \mid A \mid B \\ A & \to & aB \mid a \\ B & \to & bA \mid b \end{array}$$

Cleaning up free grammars

EXAMPLE

$$\begin{array}{rcl} S & \rightarrow & abc \mid AB \mid A \mid B \\ A & \rightarrow & aB \mid a \\ B & \rightarrow & bA \mid b \end{array}$$

- Unit production elimination

$$\begin{array}{rcl} S & \rightarrow & abc \mid AB \mid aB \mid a \mid bA \mid b \\ A & \rightarrow & aB \mid a \\ B & \rightarrow & bA \mid b \end{array}$$

Cleaning up free grammars

EXAMPLE

- From

$$\begin{array}{rcl} S & \rightarrow & ABC \mid abc \\ A & \rightarrow & aB \mid \epsilon \\ B & \rightarrow & bA \mid C \\ C & \rightarrow & \epsilon \end{array}$$

- We get

$$\begin{array}{rcl} S & \rightarrow & abc \mid AB \mid aB \mid a \mid bA \mid b \\ A & \rightarrow & aB \mid a \\ B & \rightarrow & bA \mid b \end{array}$$

Closure wrt union

LEMMA
The class of free languages is closed w.r.t. set-union.

- Meaning:
  - If $\mathcal{L}_1$ and $\mathcal{L}_2$ are free languages
  - Then $\mathcal{L}_1 \cup \mathcal{L}_2$ is a free language.

Closure wrt union

PROOF

- Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be free languages.
- Then there exist two free grammars $\mathcal{G}_1 = (V_1, T_1, S_1, \mathcal{P}_1)$ and $\mathcal{G}_2 = (V_2, T_2, S_2, \mathcal{P}_2)$ such that $\mathcal{L}_1 = \mathcal{L}(\mathcal{G}_1)$ and $\mathcal{L}_2 = \mathcal{L}(\mathcal{G}_2)$.
- Also, we can assume that $(V_1 \setminus T_1) \cap (V_2 \setminus T_2) = \emptyset$, namely that $\mathcal{G}_1$ and $\mathcal{G}_2$ do not share any non-terminal
- Let $\mathcal{G} = (V_1 \cup V_2 \cup \{S\}, T_1 \cup T_2, S, \mathcal{P}_1 \cup \mathcal{P}_2 \cup \{S \to S_1 \mid S_2\})$ where:
  - S is a new symbol not in $V_1 \cup V_2$
- Then $\mathcal{L}(\mathcal{G})$ is free and $\mathcal{L}(\mathcal{G}) = \mathcal{L}(\mathcal{G}_1) \cup \mathcal{L}(\mathcal{G}_2)$.

Closure wrt union

PROOF: Why is $\mathcal{G}$ free?

- The productions of $\mathcal{G}$ are those in $\mathcal{P}_1 \cup \mathcal{P}_2 \cup \{S \to S_1 \mid S_2\})$
- The productions in both $\mathcal{P}_1$ and $\mathcal{P}_2'$ have the form $A \to \alpha$
- The productions $S \to S_1$ and $S \to S_2$ also have the form $A \to \alpha$
- Hence $\mathcal{G}$ is free

Closure wrt union

PROOF: Why is $\mathcal{L}(\mathcal{G}) = \mathcal{L}(\mathcal{G}_1) \cup \mathcal{L}(\mathcal{G}_2)$?

- $w \in \mathcal{L}(\mathcal{G})$
- iff there exists a derivation $S \Rightarrow^* w$
- iff $S \Rightarrow S_1 \Rightarrow^* w$ or $S \Rightarrow S_2 \Rightarrow^* w$
- iff $S_1 \Rightarrow^* w$ or $S_2 \Rightarrow^* w$
- iff $w \in \mathcal{L}(\mathcal{G}_1)$ or $w \in \mathcal{L}(\mathcal{G}_2)$
- iff $w \in \mathcal{L}(\mathcal{G}_1) \cup \mathcal{L}(\mathcal{G}_2)$.

Closure wrt union

PROOF: Why insisting on $\mathcal{G}_1$ and $\mathcal{G}_2$ not sharing non-terminals?

- Take

$$\mathcal{G}_1 : \begin{array}{lcl} S_1 & \to & aA \\ A & \to & a \end{array} \qquad\qquad \mathcal{G}_2 : \begin{array}{lcl} S_2 & \to & bA \\ A & \to & b \end{array}$$

- Then $\mathcal{L}(\mathcal{G}_1) = \{aa\}$ and $\mathcal{L}(\mathcal{G}_2) = \{bb\}$.
- However, if we just put everything together and define

$$\mathcal{G} : \begin{array}{lcl} S & \to & S_1 \mid S_2 \\ S_1 & \to & aA \\ S_2 & \to & bA \\ A & \to & a \mid b \end{array}$$

- Then $\mathcal{L}(\mathcal{G}) = \{aa, ab, ba, bb\} \neq \mathcal{L}(\mathcal{G}_1) \cup \mathcal{L}(\mathcal{G}_2)$.
- What is the problem here?

Closure wrt union

PROOF: Why insisting on $\mathcal{G}_1$ and $\mathcal{G}_2$ not sharing non-terminals?

- Take

$$\mathcal{G}_1 : \begin{array}{lcl} S_1 & \to & aA \\ A & \to & a \end{array} \qquad\qquad \mathcal{G}_2 : \begin{array}{lcl} S_2 & \to & bA \\ A & \to & b \end{array}$$

- Then $\mathcal{L}(\mathcal{G}_1) = \{aa\}$ and $\mathcal{L}(\mathcal{G}_2) = \{bb\}$.
- The symbol A plays distinct roles in $\mathcal{G}_1$ and in $\mathcal{G}_2$
- To reflect this distinction, rather take

$$\mathcal{G}_1 : \begin{array}{lcl} S_1 & \to & aA \\ A & \to & a \end{array} \qquad\qquad \mathcal{G}_2 : \begin{array}{lcl} S_2 & \to & bA' \\ A' & \to & b \end{array}$$

- No mix up of productions now, and, by contruction, $\mathcal{L}(\mathcal{G}) = \{aa, bb\}$

Closure wrt concatenation

LEMMA
The class of free languages is closed w.r.t. concatenation.

- Meaning:
  - If $\mathcal{L}_1$ and $\mathcal{L}_2$ are free languages
  - Then $\{w_1 w_2 \mid w_1 \in \mathcal{L}_1 \text{ and } w_2 \in \mathcal{L}_2\}$ is a free language.

Closure wrt concatenation

PROOF

- Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be free languages.
- Then there exist two free grammars $\mathcal{G}_1 = (V_1, T_1, S_1, \mathcal{P}_1)$ and $\mathcal{G}_2 = (V_2, T_2, S_2, \mathcal{P}_2)$ such that $\mathcal{L}_1 = \mathcal{L}(\mathcal{G}_1)$ and $\mathcal{L}_2 = \mathcal{L}(\mathcal{G}_2)$.
- Also, we can assume that there is no clash between the non-terminals of $\mathcal{G}_1$ and those of $\mathcal{G}_2$
- Let $\mathcal{G} = (V_1 \cup V_2 \cup \{S\}, T_1 \cup T_2, S, \mathcal{P}_1 \cup \mathcal{P}_2 \cup \{S \rightarrow S_1 S_2\})$ where S is a new symbol not in $V_1 \cup V_2$.
- Then $\mathcal{L}(\mathcal{G})$ is free and $\mathcal{L}(\mathcal{G}) = \{w_1 w_2 \mid w_1 \in \mathcal{L}(\mathcal{G}_1) \text{ and } w_2 \in \mathcal{L}(\mathcal{G}_2)\}$.

Are free languages closed w.r.t. intersection?

Recall

LEMMA
Let $\mathcal{L}$ be a context-free language. Then there exists a context-free grammar $\mathcal{G}$ such that $\mathcal{L}(\mathcal{G}) = \mathcal{L} \setminus \{\epsilon\}$ and that has:

- No ϵ-production (i.e. no production of the shape $A \to \epsilon$)
- No unit production (i.e. no production of the shape $A \to B$)
- No useless non-terminal (i.e. non-terminals that never appear in some derivations of some strings of terminals)

NOTE
Each production $A \to \beta$ in $\mathcal{G}$ is such that either β is a single terminal or $|\beta| \geq 2$

Pumping Lemma for free languages

LEMMA
Let $\mathcal{L}$ be a free language. Then

- $\exists p \in \mathbb{N}^+$ such that
- $\forall z \in \mathcal{L}$ such that $|z| > p$
- $\exists u, v, w, x, y$ such that
  - $z = uvwxy$ and
  - $|vwx| \leq p$ and
  - $|vx| > 0$ and
  - $\forall i \in \mathbb{N}. uv^i wx^i y \in \mathcal{L}$

Pumping Lemma for free languages

PROOF

- Let $\mathcal{L}$ be a free language
- The lemma is about words longer than $p > 0$, and hence different from ϵ
- Then just consider a "cleaned-up" free grammar $\mathcal{G}$ such that $\mathcal{L}(\mathcal{G}) = \mathcal{L} \setminus \{\epsilon\}$

Pumping Lemma for free languages
PROOF

- Transform $\mathcal{G}$ in (Chomsky normal form) a grammar $\mathcal{G}'$ where each production has
  - either the form $A \to a$
  - or the form $A \to A_1 A_2$
- Example
  - $S \to aSb \mid ab$
  - Both aSb and ab are to be changed. Pick up a new non-terminal for a, say A, and a new non-terminal for b, say B, and transform
  - $S \to ASB \mid AB, \qquad A \to a, \qquad B \to b$
  - ASB is longer than 2, pick up a new non-terminal for SB, say C, and transform
  - $S \to AC \mid AB, \qquad C \to SB, \qquad A \to a, \qquad B \to b$

Pumping Lemma for free languages
PROOF

- Let k be the number of non-terminals in $\mathcal{G}'$
- Observe that any derivation tree for words in $\mathcal{L}(\mathcal{G}')$ has
  - Exactly 2^0 nodes at level 0
  - At most 2^1 nodes at level 1
  - At most 2^2 nodes at level 2
  - ......
  - At most 2^j nodes at level j
- Take $p = 2^{k+1}$

Pumping Lemma for free languages

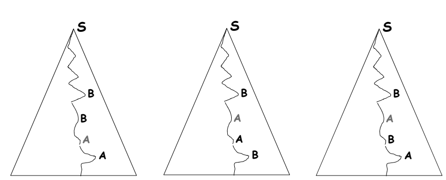
PROOF

- Let $z \in \mathcal{L}$ be such that $|z| > p$
- Then the derivation tree for z must have at least $k + 2$ levels
- Then every longest path of the derivation tree for z has a terminal at the last level and traverses at least $k + 1$ non-terminals
- Hence there is at least one non-terminal occurring twice or more along the path

Pumping Lemma for free languages

PROOF

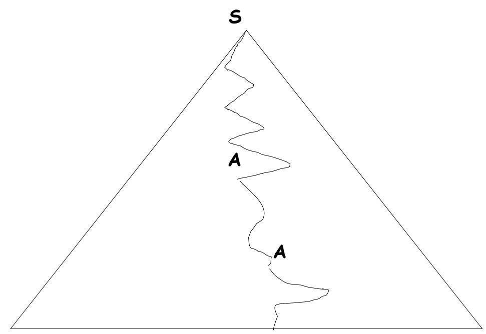
- Consider the longest path in the tree, and the **deepest pair** of occurrences of the same non-terminal along that path (i.e. choose the non-terminal whose second occurrence is found first going bottom-up)
- For instance, below the deepest pair is always the pair of As



Pumping Lemma for free languages

PROOF

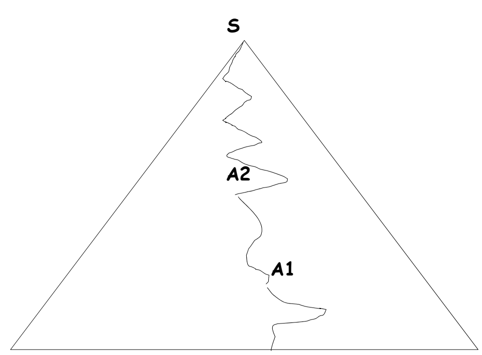
Let A be non-terminal of the deepest pair of occurrences of the same non-terminal along the path



Pumping Lemma for free languages

PROOF

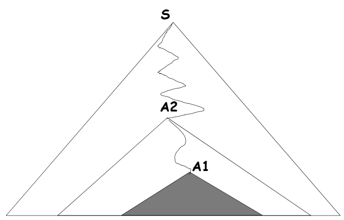
Call $A1$ and $A2$ the two occurrences of A

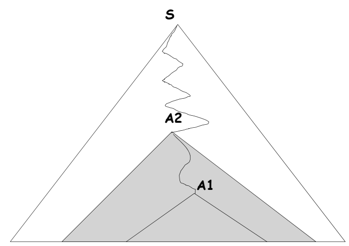


Pumping Lemma for free languages

PROOF

Then there are two distinct subtrees rooted at A: the "pink subtree" and the "green subtree"

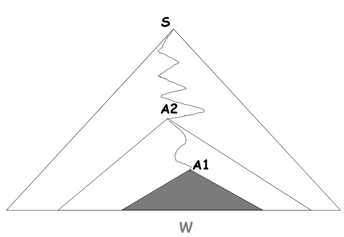


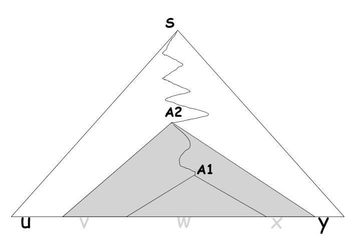


Pumping Lemma for free languages

PROOF

Then $z = uvwxy$

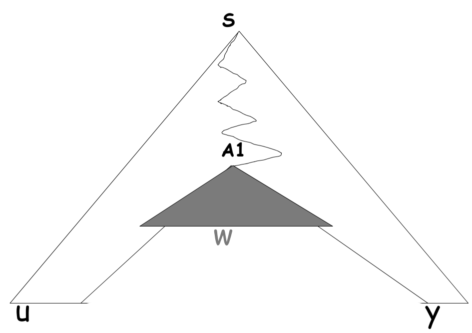




Pumping Lemma for free languages

PROOF

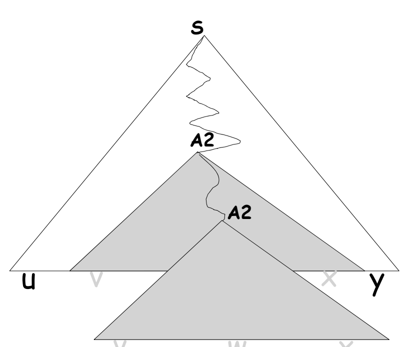
Then $uv^0wx^0y \in \mathcal{L}$.



Pumping Lemma for free languages

PROOF

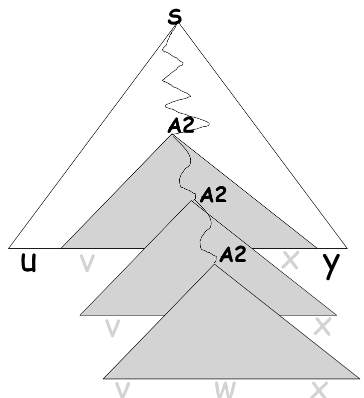
Then $uv^2wx^2y \in \mathcal{L}$.



Pumping Lemma for free languages

PROOF

Then $uv^3wx^3y \in \mathcal{L}$.



Pumping Lemma for free languages

PROOF

Then

- $\forall i \in \mathbb{N}. uv^iwx^iy \in \mathcal{L}$
- $|vwx| \leq p$
  - Because we have chosen the deepest pair of repeated non-terminals $(A1, A2)$ along the longest path from S
  - Then along the longest path from $A2$, below $A2$, no non-terminal can occur more than once
  - Then the subtree rooted at $A2$ has at most $k+1$ levels
  - Then the length of the yield of the subtree is bound by 2^{k+1}
- $|vx| > 0$
  - Because $\mathcal{G}'$ is cleaned-up
  - Then it cannot be $A \Rightarrow^+ A$ but rather $A \Rightarrow^+ \alpha A \beta$ with at least one terminal derived by either α or β

Pumping Lemma for free languages

WHAT IS THIS LEMMA GOOD FOR?

- Recall the structure of the statement
- " Let $\mathcal{L}$ be a free language. Then *PL-THESIS*."
- **By no means** the lemma can be used to show that a certain language is free
- It is used to show that a **language is not free**
- Schema of such proofs:
  - Assume that language $\mathcal{L}$ is free
  - Show that *PL-THESIS* is false, i.e., prove *not(PL-THESIS)*
  - By contradiction, conclude that $\mathcal{L}$ is not free

Pumping Lemma for free languages

PL-THESIS

- $\exists p \in \mathbb{N}^+.\ \forall z \in \mathcal{L}$: $|z| > p.\ \exists u, v, w, x, y.\ P$
- where
  - $P \equiv P1$ and $P2$ and $P3$ and $P4$
  - $P1 \equiv z = uvwxy$
  - $P2 \equiv |vwx| \leq p$
  - $P3 \equiv |vx| > 0$
  - $P4 \equiv \forall i \in \mathbb{N}.uv^i wx^i y \in \mathcal{L}$

Pumping Lemma for free languages

not(PL-THESIS)

- not ($\exists p \in \mathbb{N}^+.\ \forall z \in \mathcal{L}\colon\ |z| > p.\ \exists u, v, w, x, y.\ P$)
- $\forall p \in \mathbb{N}^+.$ not ($\forall z \in \mathcal{L}\colon\ |z| > p.\ \exists u, v, w, x, y.\ P$)
- $\forall p \in \mathbb{N}^+.\ \exists z \in \mathcal{L}\colon\ |z| > p.$ not ($\exists u, v, w, x, y.\ P$)
- $\forall p \in \mathbb{N}^+.\ \exists z \in \mathcal{L}\colon\ |z| > p.\ \forall u, v, w, x, y.$ not (P)
- $\forall p \in \mathbb{N}^+.\ \exists z \in \mathcal{L}\colon\ |z| > p.\ \forall u, v, w, x, y.$
  not ($P1$ and $P2$ and $P3$ and $P4$)

Pumping Lemma for free languages

not ($P1$ and $P2$ and $P3$ and $P4$)

- not ($P1$ and $P2$ and $P3$ and $P4$)
- not (($P1$ and $P2$ and $P3$) and $P4$)
- not ($P1$ and $P2$ and $P3$) or not $P4$
- ($P1$ and $P2$ and $P3$) implies not $P4$
- ($P1$ and $P2$ and $P3$) implies not ($\forall i \in \mathbb{N}. uv^i wx^i y \in \mathcal{L}$)
- ($P1$ and $P2$ and $P3$) implies $\exists i \in \mathbb{N}.$ not ($uv^i wx^i y \in \mathcal{L}$)

Pumping Lemma for free languages

not(PL-THESIS)

- $\forall p \in \mathbb{N}^+.\ \exists z \in \mathcal{L}\colon\ |z| > p.\ \forall u, v, w, x, y.$
  - ($z = uvwxy$ and $|vwx| \leq p$ and $|vx| > 0$)
  - implies
  - $\exists i \in \mathbb{N}. uv^i wx^i y \notin \mathcal{L}$

- Operationally:

- **Whichever** positive natural number p is
- **Choose** a word z longer than p and belonging to the language
- Show that, **whichever** unpacking of z into $uvwxy$ with $|vwx| \leq p$ and $|vx| > 0$ is taken
- A natural number i can be **chosen** which is such that $uv^i wx^i y \notin \mathcal{L}$

Pumping Lemma at work

$$\begin{array}{rrcl} \mathcal{G}: & S & \rightarrow & aSBc \mid abc \\ & cB & \rightarrow & Bc \\ & bB & \rightarrow & bb \end{array}$$

- $\mathcal{G}$ is context-dependent and $\mathcal{L}(\mathcal{G}) = \{a^n b^n c^n \mid n > 0\}$.
- Is $\mathcal{L}(\mathcal{G})$ a free language?

Pumping Lemma at work

LEMMA

$\mathcal{L} = \{a^n b^n c^n \mid n > 0\}$ is not free.

Pumping Lemma at work

PROOF

- Suppose $\mathcal{L}$ is free, and let p be an arbitrary positive integer
- Take $z = a^p b^p c^p$
- If ($z = uvwxy$ and $|vwx| \leq p$ and $|vx| > 0$)
- Then vx cannot have occurrences of both as and cs because the last occurrence of a and the first occurrence of c are $p + 1$ positions far. In fact, for some positive k and j
  - Either $vwx = a^k$
  - Or $vwx = a^k b^j$
  - Or $vwx = b^j$
  - Or $vwx = b^j c^k$
  - Or $vwx = c^k$

Pumping Lemma at work
PROOF

- Then vx has either no occurrences of c or no occurrences of a
- Then uv^0wx^0y cannot have the form $a^nb^nc^n$, hence $uv^0wx^0y \notin \mathcal{L}$
- Then, by contradiction wrt the Pumping Lemma, $\mathcal{L}$ is not free

Pumping Lemma at work
AGAIN ON THE PROOF STRUCTURE

- . . . let p be an arbitrary positive integer
  $\forall p$: Any p
- Take $z = a^pb^pc^p$
  $\exists z$: Choose $z \in \mathcal{L}$ longer than p
- . . . If ($z = uvwxy$ and $|vwx| \leq p$ and $|vx| > 0$) then ....
  $\forall u, v, w, x, y : z = uvwxy$ and $|vwx| \leq p$ and $|vx| > 0$
- . . . $uv^0wx^0y \notin \mathcal{L}$
  $\exists i$: Choose a value for the iterator
- . . . contradiction

Pumping Lemma at work

$$\begin{array}{rrcl} \mathcal{G}: & S & \rightarrow & CD \\ & C & \rightarrow & aCA \mid bCB \mid \epsilon \\ & AD & \rightarrow & aD \\ & BD & \rightarrow & bD \\ & Aa & \rightarrow & aA \\ & Ab & \rightarrow & bA \\ & Ba & \rightarrow & aB \\ & Bb & \rightarrow & bB \\ & D & \rightarrow & \epsilon \end{array}$$

- $\mathcal{G}$ is context-dependent
- What is $\mathcal{L}(\mathcal{G})$?

Pumping Lemma at work

- Derived strings have a bookmark D initially at rightmost position
$$S \rightarrow CD$$
- Strings can only grow longer by replacing C
$$C \rightarrow aCA \mid bCB \mid \epsilon$$
- Non-terminals close to the rightmost delimiter can be converted to the corresponding terminal
$$\begin{array}{rcl} AD & \rightarrow & aD \\ BD & \rightarrow & bD \end{array}$$
- Non-terminals and terminals can be swapped when the terminal is at the right of the non-terminal
$$\begin{array}{rcl} Aa & \rightarrow & aA \\ Ab & \rightarrow & bA \\ Ba & \rightarrow & aB \\ Bb & \rightarrow & bB \end{array}$$

Pumping Lemma at work

$$
\begin{array}{ll}
S & \text{by } S \to CD \\
\Rightarrow CD & \text{by } C \to aCA \\
\Rightarrow aCAD & \text{by } C \to aCA \\
\Rightarrow aaCAAD & \text{by } C \to bCB \\
\Rightarrow aabCBAAD & \text{by } C \to \epsilon \\
\Rightarrow aabBAAD & \text{by } AD \to aD \\
\Rightarrow aabBAaD & \text{by } Aa \to aA \\
\Rightarrow aabBaAD & \text{by } Ba \to aB \\
\Rightarrow aabaBAD & \text{by } AD \to aD \\
\Rightarrow aabaBaD & \text{by } Ba \to aB \\
\Rightarrow aabaaBD & \text{by } BD \to bD \\
\Rightarrow aabaabD & \text{by } D \to \epsilon \\
\Rightarrow aabaab &
\end{array}
$$

Pumping Lemma at work

$$
\mathcal{G}: \quad
\begin{array}{rcl}
S & \to & CD \\
C & \to & aCA \mid bCB \mid \epsilon \\
AD & \to & aD \\
BD & \to & bD \\
Aa & \to & aA \\
Ab & \to & bA \\
Ba & \to & aB \\
Bb & \to & bB \\
D & \to & \epsilon
\end{array}
$$

- $\mathcal{L}(\mathcal{G}) = \{ww \mid w \in \{a,b\}^*\}$
- Is $\mathcal{L}(\mathcal{G})$ a free language?

Pumping Lemma at work

LEMMA
$\mathcal{L}(\mathcal{G}) = \{ww \mid w \in \{a,b\}^*\}$ is not free.

Proof
Analogous to the previous one.
A good choice for z is $z = a^p b^p a^p b^p$.

Free or not?

- $\{a^n b^n c^n \mid n > 0\}$
- Not free, by previous lemma

- $\{a^n b^n c^j \mid n, j > 0\}$
- Free, concatenation of two free languages
- $\{a^n b^n \mid n > 0\}$ and $\{c^j \mid j > 0\}$

- $\{a^j b^n c^n \mid j, n > 0\}$
- Free, concatenation of two free languages
- $\{a^j \mid j > 0\}$ and $\{b^n c^n \mid n > 0\}$

Closure wrt intersection does not hold

LEMMA
The class of free languages is not closed w.r.t. intersection.

Proof

- By contradiction
- Take two free languages $\mathcal{L}_1$ and $\mathcal{L}_2$ whose intersection is not free
- $\mathcal{L}_1 = \{a^n b^n c^j \mid n, j > 0\}$
- $\mathcal{L}_2 = \{a^j b^n c^n \mid n, j > 0\}$
- $\mathcal{L}_1 \cap \mathcal{L}_2 = \{a^n b^n c^n \mid n > 0\}$

Training

$$\mathcal{G}: \quad S \rightarrow aSc \mid aTc \mid T$$
$$T \rightarrow bTa \mid ba$$

- Is $\mathcal{G}$ ambiguous?
- Yes
  - $S \Rightarrow aTc \Rightarrow abac$
  - $S \Rightarrow aSc \Rightarrow aTc \Rightarrow abac$
- What is $\mathcal{L}(\mathcal{G})$?

  $\{a^k b^n a^n c^k \mid k \geq 0, n > 0\}$

Training

$$\mathcal{G}: \quad \begin{aligned} S &\rightarrow 0B \mid 1A \\ A &\rightarrow 0 \mid 0S \mid 1AA \\ B &\rightarrow 1 \mid 1S \mid 0BB \end{aligned}$$

- What is $\mathcal{L}(\mathcal{G})$?

  $\{w \mid w \in \{0,1\}^* \text{ and } \#(0,w) = \#(1,w)\}$

Training

- Define $\mathcal{G}$ such that $\mathcal{L}(\mathcal{G}) = \{a^k b^n c^{2k} \mid k, n > 0\}$

  $$\begin{aligned} S &\rightarrow aScc \mid aBcc \\ B &\rightarrow bB \mid b \end{aligned}$$

- Define $\mathcal{G}$ such that $\mathcal{L}(\mathcal{G}) = \{a^k b^n c^{2k} \mid k, n \geq 0\}$

  $$\begin{aligned} S &\rightarrow aScc \mid B \\ B &\rightarrow bB \mid \epsilon \end{aligned}$$

Training

$$\begin{array}{rcl} \mathcal{G}: \quad S & \to & aBS \mid bA \\ aB & \to & Ac \mid a \\ bA & \to & S \mid Ba \end{array}$$

- Is $\mathcal{L}(\mathcal{G}) = \emptyset$?

  No: $\underline{S} \Rightarrow aB\underline{S} \Rightarrow \underline{aB}bA \Rightarrow a\underline{bA} \Rightarrow \underline{aB}a \Rightarrow aa$

Formal Languages and Compilers Paola Quaglia, 2023

Training

- Define a grammar $\mathcal{G}$ such that $\mathcal{L}(\mathcal{G})$ is the set of all the even binary numbers

  $S \quad \to \quad 0S \mid 1S \mid 0$

- Define a grammar $\mathcal{G}'$ such that $\mathcal{L}(\mathcal{G}') = \{1^n 0 \mid n \geq 0\}$

  $$\begin{array}{rcl} S & \to & A0 \mid 0 \\ A & \to & 1A \mid 1 \end{array}$$

- Is $\mathcal{L}(\mathcal{G}') = \mathcal{L}(\mathcal{G})$?

  No: 000000 is in $\mathcal{L}(\mathcal{G})$ but not in $\mathcal{L}(\mathcal{G}')$