Closure properties of CFL

Recall: We say that the family of context-free languages is **closed under the language operation** *Op* if

$$L_1, L_2, \ldots \text{ are CFL} \implies Op(L_1, L_2, \ldots) \text{ is CFL}$$

That is, if operation *Op* is applied to context-free languages, the result is also context-free.

Closure properties of CFL

Recall: We say that the family of context-free languages is **closed under the language operation** *Op* if

$$L_1, L_2, \ldots \text{ are CFL} \implies Op(L_1, L_2, \ldots) \text{ is CFL}$$

That is, if operation *Op* is applied to context-free languages, the result is also context-free.

We say the **closure is effective** if there is a mechanical procedure (=algorithm) that constructs result $Op(L_1, L_2, \ldots)$ for any context-free input languages L_1, L_2, ...

Inputs and outputs are given in the form of PDA or context-free grammar – it does not matter which format is used since both devices can be mechanically converted into each other.

Theorems. The family of CFL is effectively closed under the following operations:

- Union (if L_1, L_2 are CFL, so is $L_1 \cup L_2$),
- Concatenation (if L_1, L_2 are CFL, so is $L_1 L_2$),
- Kleene closure (if L is CFL, so is L^*),
- Substitutions with CFL (if L is CFL, and f is a substitution such that for every letter $a \in \Sigma$ language $f(a)$ is CFL, then $f(L)$ is CFL),
- Homomorphisms (if L is CFL, and h is a homomorphism then $h(L)$ is CFL),
- Inverse homomorphisms (if L is CFL, and h is a homomorphism then $h^{-1}(L)$ is CFL),
- Intersections with **regular** languages (if L is CFL and R is a regular language, then $L \cap R$ is CFL).

More theorems. The family of CFL is **NOT** closed under the following operations:

- Intersection (for some CFL L_1 and L_2 the language $L_1 \cap L_2$ is **not** CFL),
- Complementation (for some CFL L the language $\bar{L}$ is **not** CFL),
- Quotient (for some CFL L_1 and L_2 the language L_1/L_2 is **not** CFL).

Proofs.

1. **Union and concatenation**. Let L_1 be generated by grammar

$$G_1 = (V_1, T_1, P_1, S_1),$$

and L_2 by

$$G_2 = (V_2, T_2, P_2, S_2).$$

We may assume that the two variable sets V_1 and V_2 are disjoint, i.e. $V_1 \cap V_2 = \emptyset$. (If necessary we just rename some variables.)

Proofs.

1. **Union and concatenation**. Let L_1 be generated by grammar

$$G_1 = (V_1, T_1, P_1, S_1),$$

and L_2 by

$$G_2 = (V_2, T_2, P_2, S_2).$$

We may assume that the two variable sets V_1 and V_2 are disjoint, i.e. $V_1 \cap V_2 = \emptyset$. (If necessary we just rename some variables.)

Union $L_1 \cup L_2$ is generated by grammar

$$G_3 = (V_1 \cup V_2 \cup \{S_3\}, T_1 \cup T_2, P_3, S_3)$$

where S_3 is a new start symbol, and P_3 contains all productions in P_1 and P_2, and additional "initializing" productions

$$S_3 \longrightarrow S_1 \mid S_2.$$

The first derivation step by G_3 produces either S_1 or S_2, and after that a derivation by G_1 or G_2 is simulated.

Proofs.

1. **Union and concatenation**. Let L_1 be generated by grammar

$$G_1 = (V_1, T_1, P_1, S_1),$$

and L_2 by

$$G_2 = (V_2, T_2, P_2, S_2).$$

We may assume that the two variable sets V_1 and V_2 are disjoint, i.e. $V_1 \cap V_2 = \emptyset$. (If necessary we just rename some variables.)

Concatenation $L_1 L_2$ is generated by grammar

$$G_4 = (V_1 \cup V_2 \cup \{S_4\}, T_1 \cup T_2, P_4, S_4)$$

where S_4 is a new start symbol, and P_4 contains all productions in P_1 and P_2, and additional initializing production

$$S_4 \longrightarrow \ S_1 S_2.$$

After first derivation step the sentential form is $S_1 S_2$, and from S_1 and S_2 we can derive words of L_1 and L_2, respectively.

2. **Kleene ***. Let L be generated by grammar

$$G = (V, T, P, S).$$

Then L^* is generated by grammar

$$G' = (V \cup \{S'\}, T, P', S')$$

where S' is a new start symbol, and P' contains all productions in P plus productions

$$S' \longrightarrow SS' \mid \varepsilon.$$

From S' one gan generate an arbitrarily long sequence of S's:

$$S' \Rightarrow SS' \Rightarrow \cdots \Rightarrow SS \ldots SS' \Rightarrow SS \ldots S,$$

and each S produces a word of L.

3. **Substitution**. Let $L \subseteq \Sigma^*$ be generated by grammar

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

and for each $a \in \Sigma$ let $f(a)$ be generated by grammar

$$G_a = (V_a, T_a, P_a, S_a).$$

Assume that all variable sets V and V_a are disjoint.

3. **Substitution**. Let $L \subseteq \Sigma^*$ be generated by grammar

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

and for each $a \in \Sigma$ let $f(a)$ be generated by grammar

$$G_a = (V_a, T_a, P_a, S_a).$$

Assume that all variable sets V and V_a are disjoint.

The following grammar generates $f(L)$:

$$G' = (V', T', P', S),$$

where

$$V' = V \cup \left(\bigcup_{a \in \Sigma} V_a \right)$$

contains all variables of all grammars G and G_a,

$$T' = \bigcup_{a \in \Sigma} T_a$$

contains all terminals of grammars G_a.

Productions P' include all productions that are in P_a's, plus for every production in P a production obtained by replacing all occurences of terminals a by the start symbol S_a of the corresponding grammar G_a.

Example. Let $L = L(G)$ be generated by

$$G = (\{X\}, \{a, b\}, P, X)$$

with productions

$$X \longrightarrow aXb \mid bXa \mid \varepsilon.$$

Let f be a substitution such that

- the language $f(a)$ is generated by $G_a = (\{Y\}, \{0, 1\}, P_a, Y)$ with productions

$$Y \longrightarrow 0YY0 \mid 1,$$

- the language $f(b)$ is generated by $G_b = (\{Z\}, \{0, 1\}, P_b, Z)$ with productions

$$Z \longrightarrow 0Z1Z \mid \varepsilon.$$

Then $f(L)$ is generated by the grammar $G' = (V', T', P', S')$ where

$$\begin{aligned} V' &= \\ T' &= \\ S' &= \\ P' &= \end{aligned}$$

4. **Homomorphism**. Follows from previous proof because homomorphism is a special type of substitution.

4. **Homomorphism**. Follows from previous proof because homomorphism is a special type of substitution.

We can also make a direct construction where we replace in the productions all occurrences of terminal letters by their homomorphic images.

Example. If L is generated by a grammar with productions

$$\begin{aligned} E &\longrightarrow E + E \mid N \\ N &\longrightarrow 0N \mid 1N \mid 0 \mid 1 \end{aligned}$$

and homomorphism h is given by $h(+) = \times$, $h(0) = ab$, $h(1) = ba$ then $h(L)$ is generated by

$$\begin{aligned} E &\longrightarrow \\ N &\longrightarrow \end{aligned}$$

4. **Homomorphism**. Follows from previous proof because homomorphism is a special type of substitution.

We can also make a direct construction where we replace in the productions all occurrences of terminal letters by their homomorphic images.

Example. If L is generated by a grammar with productions

$$\begin{aligned} E &\longrightarrow E + E \mid N \\ N &\longrightarrow 0N \mid 1N \mid 0 \mid 1 \end{aligned}$$

and homomorphism h is given by $h(+) = \times$, $h(0) = ab$, $h(1) = ba$ then $h(L)$ is generated by

$$\begin{aligned} E &\longrightarrow E \times E \mid N \\ N &\longrightarrow abN \mid baN \mid ab \mid ba \end{aligned}$$

5. **Inverse homomorphism**. In this case it is easier to use PDA instead of grammars. Let L be recognized by the PDA

$$M = (Q, \Delta, \Gamma, \delta, q_0, Z_0, F)$$

by final state, and let

$$h : \Sigma \longrightarrow \Delta^*$$

be a homomorphism. Let us construct a PDA M' that recognizes the language $h^{-1}(L)$ by final state, that is, it accepts all words w such that $h(w) \in L$.

5. **Inverse homomorphism**. In this case it is easier to use PDA instead of grammars. Let L be recognized by the PDA

$$M = (Q, \Delta, \Gamma, \delta, q_0, Z_0, F)$$

by final state, and let

$$h : \Sigma \longrightarrow \Delta^*$$

be a homomorphism. Let us construct a PDA M' that recognizes the language $h^{-1}(L)$ by final state, that is, it accepts all words w such that $h(w) \in L$.

Idea: On an input w, machine M' computes $h(w)$ and simulates M with input $h(w)$. If $h(w)$ is accepted by M then w is accepted by M'.

The result of h on w is stored on a buffer, or **"virtual input tape"**, inside the control unit. The simulation of M reads its input from this virtual tape. $h(w)$ is computed from w letter-by-letter as needed: as soon as the virtual tape becomes empty the next "real" input letter a is scanned, and $h(a)$ is added on the virtual tape.

In detail: Let

$$B = \{u \in \Delta^* \mid \ u \text{ is a suffix of } h(a) \text{ for some } a \in \Sigma\}.$$

Set B consists of all possible contents of the "virtual" input tape. Set B is of course a finite set.

Let $M = (Q, \Delta, \Gamma, \delta, q_0, Z_0, F)$ recognize L. For $h^{-1}(L)$ we construct the PDA

$$M' = (Q', \Sigma, \Gamma, \delta', q_0', Z_0, F'),$$

where

$$\begin{aligned} Q' &= Q \times B, \\ q_0' &= [q_0, \varepsilon], \text{ and} \\ F' &= F \times \{\varepsilon\}. \end{aligned}$$

δ' contains two types of productions:

- For every transition of the original machine M we have the simulating transition in M': If
$$(p, \gamma) \in \delta(q, a, Z)$$
then
$$([p, x], \gamma) \in \delta'([q, ax], \varepsilon, Z)$$
for every $ax \in B$. Here $a \in \Sigma \cup \{\varepsilon\}$. Notice that this is an ε-move: M' reads the input letter a from the virtual tape.

- When the virtual tape becomes empty, we have a transition for reading the next real input letter a, and loading its homomorphic image $h(a)$ to the virtual tape: For all $q \in Q$, $a \in \Sigma$, and $Z \in \Gamma$ we have the transition
$$([q, h(a)], Z) \in \delta'([q, \varepsilon], a, Z).$$

Initially the virtual tape is empty, so the initial state of M' is

$$[q_0, \varepsilon].$$

The input word is accepted if in the end the virtual input tape is empty (M has consumed the whole input word), and the state of M is a final state. Therefore the final states of M' are elements of

$$F \times \{\varepsilon\}.$$

Example. Consider the PDA

$$M = (\{q\}, \{a, b\}, \{A\}, \delta, q, A, \{q\})$$

where

$$\begin{aligned} \delta(q, a, A) &= \{(q, AA)\} \\ \delta(q, b, A) &= \{(q, \varepsilon)\} \end{aligned}$$

and let h be the homomorphism $h(0) = b$, $h(1) = aa$. Let us construct a PDA for $h^{-1}(L(M))$:

Possible contents of the virtual tape are $B =$

so the state set of the new PDA will be $Q' =$

The start state is $S' =$

and there is only one final state $F' =$

Example. Consider the PDA

$$M = (\{q\}, \{a, b\}, \{A\}, \delta, q, A, \{q\})$$

where

$$\begin{aligned} \delta(q, a, A) &= \{(q, AA)\} \\ \delta(q, b, A) &= \{(q, \varepsilon)\} \end{aligned}$$

and let h be the homomorphism $h(0) = b$, $h(1) = aa$. Let us construct a PDA for $h^{-1}(L(M))$:

Possible contents of the virtual tape are $B = \{b, \varepsilon, aa, a\}$

so the state set of the new PDA will be $Q' = \{[q, b], [q, \varepsilon], [q, aa], [q, a]\}$

The start state is $S' = [q, \varepsilon]$

and there is only one final state $F' = \{[q, \varepsilon]\}$

Example. Consider the PDA

$$M = (\{q\}, \{a, b\}, \{A\}, \delta, q, A, \{q\})$$

where

$$\begin{aligned} \delta(q, a, A) &= \{(q, AA)\} \\ \delta(q, b, A) &= \{(q, \varepsilon)\} \end{aligned}$$

and let h be the homomorphism $h(0) = b$, $h(1) = aa$. Let us construct a PDA for $h^{-1}(L(M))$:

Possible contents of the virtual tape are $B = \{b, \varepsilon, aa, a\}$

so the state set of the new PDA will be $Q' = \{[q, b], [q, \varepsilon], [q, aa], [q, a]\}$

The start state is $S' = [q, \varepsilon]$

and there is only one final state $F' = \{[q, \varepsilon]\}$

From the original transition $(q, AA) \in \delta(q, a, A)$ we get two simulating transitions, and from $(q, \varepsilon) \in \delta(q, b, A)$ one transition:

Finally, we have transitions for loading the virtual tape:

Example. Let us show that the language

$$L = \{a^n b^{2n} c^{3n} \mid n \geq 1\}$$

is not context-free.

6. **NOT closed under Intersection**. It is enough to find one counter example.

6. **NOT closed under Intersection**. It is enough to find one counter example. Let

$$L_1 = \{a^n b^n c^m \mid n, m \geq 1\}$$

and

$$L_2 = \{a^m b^n c^n \mid n, m \geq 1\}.$$

Both L_1 and L_2 are context-free: L_1 is concatenation of context-free languages

$$\{a^n b^n \mid n \geq 1\} \text{ and } c^+,$$

and L_2 is concatenation of

$$a^+ \text{ and } \{b^n c^n \mid n \geq 1\}.$$

(We may also directly construct grammars for both L_1 and L_2.)

But even though L_1 and L_2 are context-free, their intersection

$L_1 \cap L_2 =$

is not. So the family of CFL is not closed under intersection.

6. **NOT closed under Intersection**. It is enough to find one counter example. Let

$$L_1 = \{a^n b^n c^m \mid n, m \geq 1\}$$

and

$$L_2 = \{a^m b^n c^n \mid n, m \geq 1\}.$$

Both L_1 and L_2 are context-free: L_1 is concatenation of context-free languages

$$\{a^n b^n \mid n \geq 1\} \text{ and } c^+,$$

and L_2 is concatenation of

$$a^+ \text{ and } \{b^n c^n \mid n \geq 1\}.$$

(We may also directly construct grammars for both L_1 and L_2.)

But even though L_1 and L_2 are context-free, their intersection

$$L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 1\}$$

is not. So the family of CFL is not closed under intersection.

7. **NOT closed under complementation**. If they were, then the family of CFL would be closed under intersection as well:

$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}},$$

a contradiction with 6 above.

8. **NOT closed under quotients (with context-free languages)**. Homework.

8. **NOT closed under quotients (with context-free languages)**. Homework.

An easy to prove variant: there exists a context-free language L and some (non-context-free) language K such that L/K is not context-free. Choose

$$L = \{a^n b^m c^k \# d^k e^m f^n \mid n, m, k \geq 1\},$$

$$K = \{\# d^n e^n f^n \mid n \geq 1\}.$$

Here L is context-free but

$L/K =$

is not context-free.

8. **NOT closed under quotients (with context-free languages)**. Homework.

An easy to prove variant: there exists a context-free language L and some (non-context-free) language K such that L/K is not context-free. Choose

$$L = \{a^n b^m c^k \# d^k e^m f^n \mid n, m, k \geq 1\},$$

$$K = \{\# d^n e^n f^n \mid n \geq 1\}.$$

Here L is context-free but

$L/K = \{a^n b^n c^n \mid n \geq 1\}$

is not context-free.

9. **Intersection with regular languages**. Even though the intersection of two context-free languages may be non-context-free, the intersection of any context-free language with a regular language is always context-free.

Proof.

9. **Intersection with regular languages**. Even though the intersection of two context-free languages may be non-context-free, the intersection of any context-free language with a regular language is always context-free.

Proof. Let L be recognized by the PDA

$$M = (Q_M, \Sigma, \Gamma, \delta_M, q_M, Z_0, F_M)$$

by final state, and let R be recognized by the DFA

$$A = (Q_A, \Sigma, \delta_A, q_A, F_A).$$

We construct a PDA M' that runs both M and A in parallel. The state set of M' is

$$Q' = Q_M \times Q_A.$$

The states are used to store the states of both M and A for the input that has been read so far.

9. **Intersection with regular languages**. Even though the intersection of two context-free languages may be non-context-free, the intersection of any context-free language with a regular language is always context-free.

Proof. Let L be recognized by the PDA

$$M = (Q_M, \Sigma, \Gamma, \delta_M, q_M, Z_0, F_M)$$

by final state, and let R be recognized by the DFA

$$A = (Q_A, \Sigma, \delta_A, q_A, F_A).$$

We construct a PDA M' that runs both M and A in parallel. The state set of M' is

$$Q' = Q_M \times Q_A.$$

The states are used to store the states of both M and A for the input that has been read so far.

All ϵ-transitions of M are simulated as such, without changing the Q_A-component of the state. But whenever M reads next input letter, the Q_A-component is changed according to DFA A. Clearly, if in the end of the input word both Q_M- and Q_A-components are final states, the word is accepted by both M and A.

$$\begin{aligned} M &= (Q_M, \Sigma, \Gamma, \delta_M, q_M, Z_0, F_M) \\ A &= (Q_A, \Sigma, \delta_A, q_A, F_A) \end{aligned}$$

More precisely, we construct the PDA

$$M' = (Q_M \times Q_A, \Sigma, \Gamma, \delta', [q_M, q_A], Z_0, F_M \times F_A)$$

where δ' is described below:

Each transition

$$(q', \gamma) \in \delta_M(q, a, Z)$$

by the original PDA, and each state $p \in Q_A$ of the DFA provide the transition

$$([q', \delta_A(p, a)], \gamma) \in \delta'([q, p], a, Z)$$

in the new machine M'.

(The first state component and the stack follow PDA M. The second state component simulates DFA A: On input a state p is changed to $\delta_A(p, a)$. Note that if $a = \varepsilon$ then $\delta_A(p, a) = p$.)

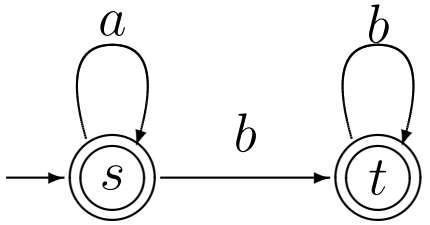
Example. Consider the PDA

$$M = (\{q\}, \{a, b\}, \{A\}, \delta, q, A, \{q\})$$

where

$$\begin{aligned} \delta(q, a, A) &= \{(q, AA)\} \\ \delta(q, b, A) &= \{(q, \varepsilon)\} \end{aligned}$$

and let $R = a^*b^*$. Then R is recognized by the DFA



The intersection $L(M) \cap R$ is recognized by the PDA

$$M' = (Q', \{a, b\}, \{A\}, \delta', q_0, A, F)$$

where

$$\begin{aligned} Q' &= \\ q_0 &= \\ F &= \end{aligned}$$

and δ' contains transitions

$$\begin{aligned} \delta'([q, s], a, A) &= \\ \delta'([q, s], b, A) &= \\ \delta'([q, t], a, A) &= \\ \delta'([q, t], b, A) &= \end{aligned}$$

Example. Let us prove that

$$L = \{0^n 1 0^n 1 0^n \mid n \geq 1\}$$

is not context-free.

Example. Let us prove that

$$L = \{0^n 1 0^n 1 0^n \mid n \geq 1\}$$

is not context-free.

Assume it is a CFL. Define a substitution $f(0) = \{a, b, c\}$, $f(1) = \{\#\}$. Since we assumed that L is context-free, so is

$L_1 = f(L) \cap (a^+ \# b^+ \# c^+) =$

Example. Let us prove that

$$L = \{0^n 1 0^n 1 0^n \mid n \geq 1\}$$

is not context-free.

Assume it is a CFL. Define a substitution $f(0) = \{a, b, c\}$, $f(1) = \{\#\}$. Since we assumed that L is context-free, so is

$$L_1 = f(L) \cap (a^+ \# b^+ \# c^+) = \{a^n \# b^n \# c^n \mid n \geq 1\}$$

Example. Let us prove that

$$L = \{0^n 1 0^n 1 0^n \mid n \geq 1\}$$

is not context-free.

Assume it is a CFL. Define a substitution $f(0) = \{a, b, c\}$, $f(1) = \{\#\}$. Since we assumed that L is context-free, so is

$$L_1 = f(L) \cap (a^+ \# b^+ \# c^+) = \{a^n \# b^n \# c^n \mid n \geq 1\}$$

Define then a homomorphism $h(a) = a$, $h(b) = b$, $h(c) = c$, $h(\#) = \varepsilon$. Then the language

$$L_2 = h(L_1) =$$

is context-free.

Example. Let us prove that

$$L = \{0^n 1 0^n 1 0^n \mid n \geq 1\}$$

is not context-free.

Assume it is a CFL. Define a substitution $f(0) = \{a, b, c\}$, $f(1) = \{\#\}$. Since we assumed that L is context-free, so is

$$L_1 = f(L) \cap (a^+ \# b^+ \# c^+) = \{a^n \# b^n \# c^n \mid n \geq 1\}$$

Define then a homomorphism $h(a) = a$, $h(b) = b$, $h(c) = c$, $h(\#) = \varepsilon$. Then the language

$$L_2 = h(L_1) = \{a^n b^n c^n \mid n \geq 1\}$$

is context-free.

But we know that L_2 is not context-free, a contradiction. Our initial assumption that L is a CFL, must be incorrect.

Example. Let us show that the language

$$L = \{ww \mid w \in \{a, b\}^*\}$$

is not context-free.

Example. Let us show that the language

$$L = \{ww \mid w \in \{a, b\}^*\}$$

is not context-free.

If it were CFL, so would be

$L_1 = L \cap (a^+b^+a^+b^+) =$

Example. Let us show that the language

$$L = \{ww \mid w \in \{a, b\}^*\}$$

is not context-free.

If it were CFL, so would be

$$L_1 = L \cap (a^+b^+a^+b^+) = \{a^m b^k a^m b^k \mid m, k \geq 1\}$$

Example. Let us show that the language

$$L = \{ww \mid w \in \{a, b\}^*\}$$

is not context-free.

If it were CFL, so would be

$$L_1 = L \cap (a^+b^+a^+b^+) = \{a^m b^k a^m b^k \mid m, k \geq 1\}$$

Define then substitution $f(a) = \{a, c\}$, $f(b) = \{b, d\}$. If L_1 is a CFL so would be

$$L_2 = f(L_1) \cap (a^+b^+c^+d^+) =$$

Example. Let us show that the language

$$L = \{ww \mid w \in \{a, b\}^*\}$$

is not context-free.

If it were CFL, so would be

$$L_1 = L \cap (a^+b^+a^+b^+) = \{a^m b^k a^m b^k \mid m, k \geq 1\}$$

Define then substitution $f(a) = \{a, c\}$, $f(b) = \{b, d\}$. If L_1 is a CFL so would be

$$L_2 = f(L_1) \cap (a^+b^+c^+d^+) = \{a^m b^k c^m d^k \mid m, k \geq 1\}$$

But we have proved earlier using the pumping lemma that L_2 is not context-free, so the original language L is not context-free either.