Example

Consider $L = \{0^i 1^i 2^i \mid i \geqslant 1\}$, and let n be the pumping lemma constant associated with L. We choose $z = 0^n 1^n 2^n$

For any factorization of z into $uvwxy$, with $|vwx| \leqslant n$ and v and x not both empty, we have that vwx cannot contain both 0 and 2, because the rightmost 0 and the leftmost 2 are $n+1$ places away one from the other

We therefore have the following cases:

- vwx does not contain 2; then vx has only 0 and 1; then uwy, which should be in L, has n occurrences of 2 but less than n occurrences of 0 or 1
- vwx does not contain 0; a similar reasoning as in the first case applies

Consequences of the pumping lemma

A CFL cannot generate **crossing pairs**

Example : $L = \{0^i 1^j 2^i 3^j \mid i, j \geqslant 1\}$

Given n, we choose $z = 0^n 1^n 2^n 3^n$. Then vwx covers occurrences of at most two alphabet symbols. In all possible fatorizations, the strings generated by iteration do not belong to L

Consequences of the pumping lemma

A CFL cannot generate **string copies**

Example : $L = \{ww \mid w \in \{0,1\}^*\}$

Given n, we choose $z = 0^n1^n0^n1^n$. In all possible fatorizations, the strings generated by iteration do not belong to L

Exercise

Using the pumping lemma, prove that the language

$$L = \{a^i b^j c^k \mid i, j \geqslant 0,\ k = \max\{i, j\}\}$$

is not context-free

Exercise

Solution Let us assume that L is a CFL; we will establish a contradiction. Let n be the pumping lemma constant associated with L

We choose $z = a^n b^n c^n \in L$ and analyze all possible factorizations $z = uvwxy$ with $|vwx| \leqslant n$ and $vx \neq \epsilon$, looking for a factorization that satisfies the pumping lemma

Exercise

$$z \;=\; \underbrace{a\cdots\cdots a}_{a\text{ block}}\underbrace{b\cdots\cdots b}_{b\text{ block}}\underbrace{c\cdots\cdots c}_{c\text{ block}}$$

We distinguish the following cases

- vwx is placed into the a block or into the b block
- vwx is placed into the c block
- vwx is placed across the a and b blocks, or else across the b and c blocks
  - v or x contain both a and b, or both b and c
  - v is placed into the a block and x is placed into the b block
  - v is placed into the b block and x is placed into the c block

Exercise

vwx is placed into the a block : consider the new string uv^kwx^ky with $k > 1$, which must belong to L

$\#_a$ (the number of a's) increases ($> n$), since $vx \neq \epsilon$, while $\#_c$ remains unchanged ($= n$) and equal to $\#_b$, that is, the minimum between $\#_a$ and $\#_b$

We therefore conclude that $uv^kwx^ky \notin L$ for $k > 1$

A similar reasoning applies to the case where vwx is placed into the b block

Exercise

vwx is placed into the c block : consider the new string uv^kwx^ky with $k = 0$, which must belong to L

$\#_c$ decreases ($< n$), since $vx \neq \epsilon$, and is no longer equal to the maximum between $\#_a$ $\#_b$, which is n, since the a block and the b block both remain unchanged

We therefore conclude that $uv^kwx^ky \notin L$ for $k = 0$

Exercise

vwx is placed across the a and b blocks or else across the b and c blocks

- v or x include both a and b : choosing $k = 2$, we break the structure $a^*b^*c^*$ and the new string doesn't belong to L
- v or x include both b and c : we use the same argument of the previous point
- v is placed into the a block and x is placed into the b block : choosing $k = 2$, increases $\#_a$ and/or $\#_b$ $(> n)$, while $\#_c$ remains unchanged $(= n)$ and therefore will not be equal to the maximum required; therefore the new string does not belong to L

Exercise

vwx is placed across the a and b blocks or else across the b and c blocks (continued)

- v is placed into the b block and x is placed into the c block
    - if $x \neq \epsilon$ we choose $k = 0$; $\#_c$ becomes smaller (and so does $\#_b$ if $v \neq \epsilon$) but $\#_a$ does not change, and provides the maximum value; therefore $uv^k wx^k y \notin L$ for $k = 0$
    - if $x = \epsilon$ we choose $k > 1$ so that $\#_b$ gets larger than $\#_a$, and $\#_c$ does not change; therefore $uv^k wx^k y \notin L$ for some appropriate k

Exercise

In none of the possible cases we have been able to satisfy the pumping lemma: we have established a **contradiction**

We then conclude that language L is not CFL

Substitution

Assume two (finite) alphabets Σ and Δ, and a function

$$s : \Sigma \rightarrow 2^{\Delta^*}$$

any subset of Δ^*

Let $w \in \Sigma^*$, with $w = a_1 a_2 \cdots a_n$, $a_i \in \Sigma$. We define

$$s(w) = s(a_1).s(a_2).\cdots .s(a_n)$$

and, for $L \subseteq \Sigma^*$, we define

$$s(L) = \bigcup_{w \in L} s(w)$$

Function s is called a **substitution**

Example

Let $s(0) = \{a^n b^n \mid n \geqslant 1\}$ and $s(1) = \{aa, bb\}$

Then $s(01)$ is a language whose strings have the form $a^n b^n aa$ or $a^n b^{n+2}$, with $n \geqslant 1$

Let $L = L(\mathbf{0}^*)$. Then $s(L)$ is a language whose strings have the form

$$a^{n_1} b^{n_1} a^{n_2} b^{n_2} \cdots a^{n_k} b^{n_k},$$

with $k \geqslant 0$ and with $n_1, n_2, \ldots, n_k$ positive integers

Substitution

Next theorem is used later to prove several closure properties for CFL in a unified way and through very simple proofs

Theorem Let L be a CFL defined over Σ and let s be a substitution defined on Σ such that, for each $a \in \Sigma$, $s(a)$ is a CFL. Then $s(L)$ is a CFL

Proof Let $G = (V, \Sigma, P, S)$ be a CFG generating L and, for each $a \in \Sigma$, let $G_a = (V_a, T_a, P_a, S_a)$ be a CFG generating $s(a)$

Substitution

We construct a CFG $G' = (V', T', P', S)$ with

$$V' = (\bigcup_{a \in \Sigma} V_a) \cup V$$
$$T' = \bigcup_{a \in \Sigma} T_a$$
$$P' = (\bigcup_{a \in \Sigma} P_a) \cup P_R$$

where P_R is obtained from P by replacing each occurrence of a in any right-hand side with symbol S_a

Substitution

We prove $L(G') = s(L)$

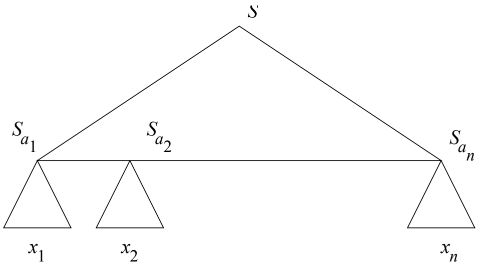
(Part $\supseteq$) Let $w \in s(L)$. Then there exists a string $x \in L$ such that

$$x = a_1 a_2 \cdots a_n$$

Furthermore, there exist strings $x_i \in s(a_i)$, such that $w = x_1 x_2 \cdots x_n$

Substitution

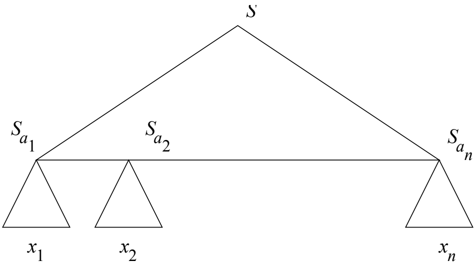
The associated parse tree for G' must have the form



We can then generate $S_{a_1} S_{a_2} \cdots S_{a_n}$ in G', and then generate $x_1 x_2 \cdots x_n = w$. Therefore $w \in L(G')$

Substitution

(Part $\subseteq$) Let $w \in L(G')$. Then the parse tree for w must have the form



Substitution

We can remove the subtrees at the bottom, and get a parse tree with yield

$$S_{a_1} S_{a_2} \cdots S_{a_n}$$

corresponding to a string $a_1 a_2 \cdots a_n \in L(G)$

We must also have $w \in s(a_1 a_2 \cdots a_n)$, and thus $w \in s(L)$ $\square$

Applications of the substitution theorem

Theorem The CFLs are closed under the following operations

- union
- concatenation
- Kleene closure $(*)$ and positive closure $(+)$
- homomorphism

Proof For each of the operators above, we define a specific substitution and we apply the previous theorem

Union : Given two CFLs L_1 and L_2, consider the CFL $L = \{1, 2\}$. and define $s(1) = L_1$, $s(2) = L_2$. We have $L_1 \cup L_2 = s(L)$, which still is a CFL

Applications of the substitution theorem

Concatenation : Given two CFLs L_1 and L_2, consider the CFL $L = \{1.2\}$ and define $s(1) = L_1$, $s(2) = L_2$. We thus have $L_1.L_2 = s(L)$, which still is a CFL

$*$ *and* $+$ *closures* : Given a CFL L_1, consider the CFL $L = \{1\}^*$ and define $s(1) = L_1$. We have $L_1^* = s(L)$, which still is a CFL. A similar argument holds for $+$

Homomorphism : Assume a CFL L and a homomorphism h, both over Σ. We define $s(a) = \{h(a)\}$ for each $a \in \Sigma$. We then have $h(L) = s(L)$, which still is a CFL □