XI

ELEVENTH LECTURE OF
AUTOMATA & FORMAL LANGUAGES
30 October 2023

CHAPTER 3 CONTEXT-FREE LANGUAGES

Lecture III
page 7

The CHOMSKY hierarchy

Let $\alpha \longrightarrow \beta$ be a production rule.

noncontracting (nc): if $|\alpha| \leq |\beta|$

context-sensitive (c-s): if $\alpha = \gamma A \delta$, $\beta = \gamma \eta \delta$ with $\gamma, \delta, \eta \in \Omega^*$, $A \in V$, $|\eta| \geq 1$

context-free (c-f): if $\alpha = A$, $A \in V$, $|\beta| \geq 1$

regular (reg): if $\alpha = A$ and $\begin{pmatrix} \beta = a \text{ or} \\ \beta = aB \end{pmatrix}$ for $A, B \in V$, $a \in \Sigma$

CONTEXT-FREE RULE

$$A \longrightarrow \beta \quad \text{where } \beta \neq \varepsilon,\ \beta \in \Omega^*.$$

Example: $L = \{0^n 1^n ; n > 0\}$

Grammar: $S \longrightarrow 0S1$, $S \longrightarrow 01$

Not regular due to (R)PL.

First goal

UNDERSTAND CONTEXT-FREE DERIVATIONS

so that we can prove things about them!

Observations

If G is noncontracting, $|\Omega| = k$, $|w| = n$ — Shortest G-derivation of w is bounded by $n \cdot k^n$

Lemma 1.18 in the typed notes

If G is regular, $|\Omega| = k$, $|w| = n$ — Any G-derivation of w has length n

Lemma 2.1 in the typed notes

Most importantly, the bound does not depend on k.

Look at the following c-f example:

$V = \{S, A_1, \dots, A_{k-1}\}$ $\quad |V| = k$

$S \longrightarrow A_1$

$A_{k-1} \longrightarrow aS$

$A_{k-1} \longrightarrow a$

$A_i \longrightarrow A_{i+1}$ for $0 < i < k-1$

This grammar accepts a^n, but takes $k \cdot n$ steps to do so.

In general, c-f grammar do not allow for a bound independent of k.

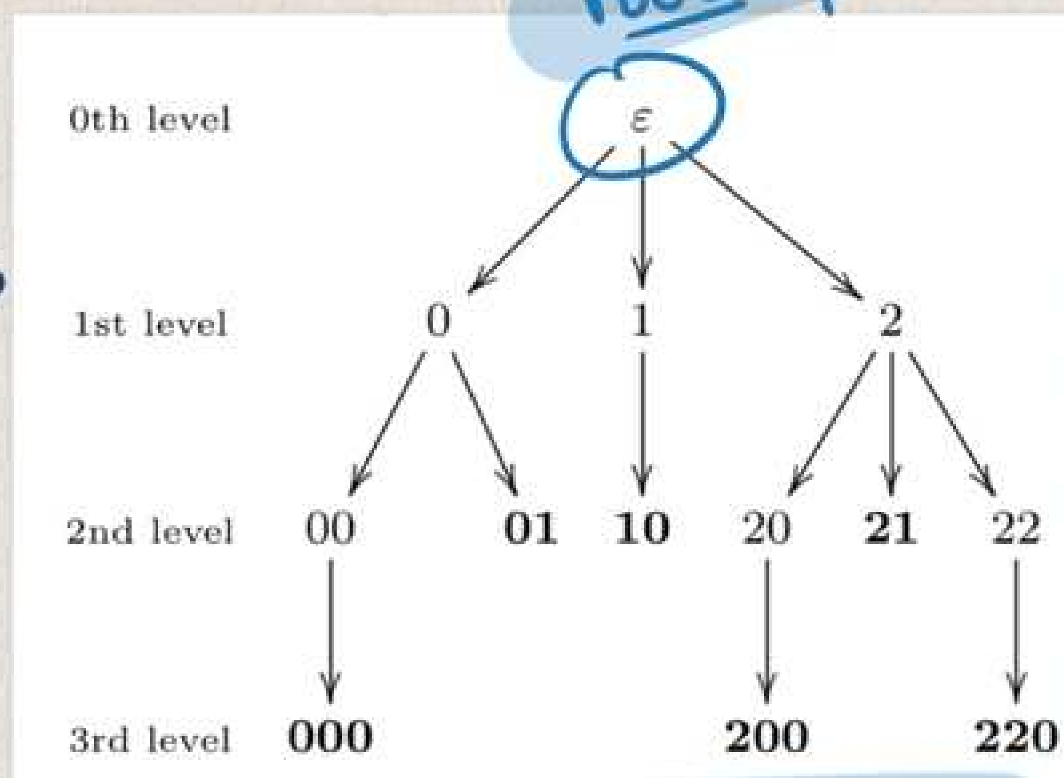
TREES

In my slides, trees grow downwards.
[This is just a drawing convention]

We consider trees that are finitely branching.

Our trees are going to subsets of $\mathbb{N}^*$:

root of the tree



finitely branching tree

<u>Definition</u> $T \subseteq \mathbb{N}^*$ is a finitely branching tree if it is closed under initial segments ($s \in T$ & $t \subseteq s \Longrightarrow t \in T$) and for each $t \in T$ there is $n \in \mathbb{N}$ s.t.

$$tk \in T \iff k < n.$$

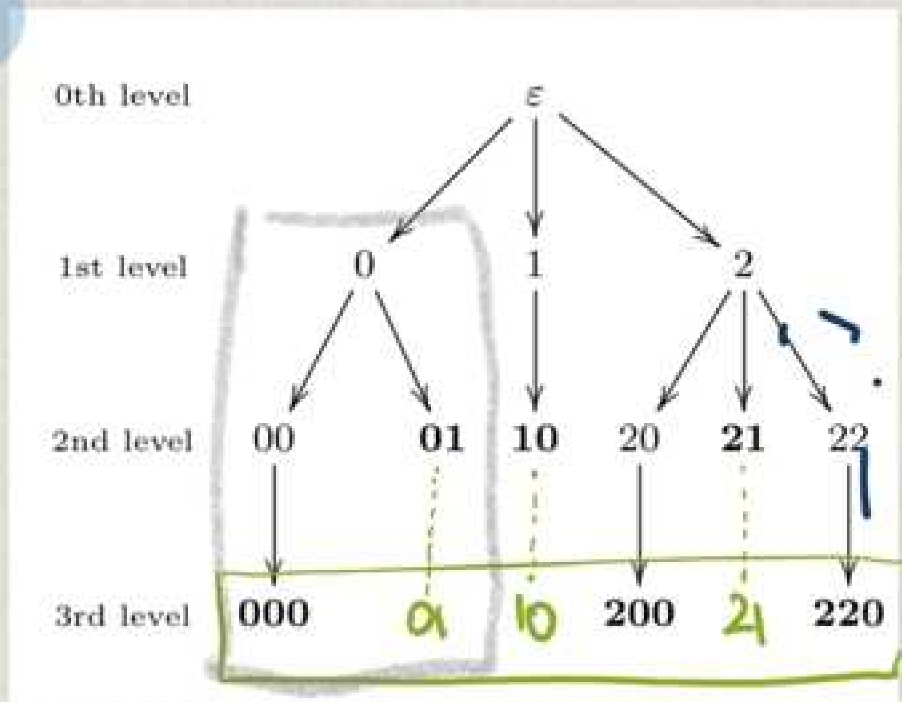
We say t is n-branching or has n successors in this case.

If t has no successors, we call it a leaf.

t is on kth level if $|t| = k$.

If $t \in T$, the **branch of T to t** is

$$\{t \upharpoonright k \,;\, k \leq |t|\}$$



Note: This is not an order on the entire tree: if $t \subsetneq s$, then neither $t \not< s$ nor $s \not< t$

If T is a tree, we can define a partial order $<$ called the *left-to-right order* as follows:

$$s < t :\iff s \neq t \text{ and if } k \text{ is least such that } s(k) \neq t(k), \text{ then } s(k) < t(k).$$

If X is a set of nodes on the same level of a tree T, then $<$ is a total order on X; similarly, if X is a set of leaves of T, then $<$ is a total order on X. In particular, the leaves of a tree are totally ordered from left to right via the order $<$.

SUBTREE

If $t \in T$, we define the **subtree of T at t** to be

$$T_t := \{s \,;\, ts \in T\}$$

LABELLING

If Ω is any set, we call a function

$$\ell : T \longrightarrow \Omega$$

an **Ω-labelling**.

DEFINITION:
PARSE TREE

(T, ℓ) : tree T with Ω-labelling ℓ

If $G = (\Sigma, V, P, S)$ is a context-free grammar and $A \in V$, we say that a pair $\mathbf{T} := (T, \ell)$ is a *G-parse tree starting from A* if T is a finite finitely branching tree and $\ell : T \to \Omega$ is a function satisfying

(a) $\ell(\varepsilon) = A$,

"starting from A"

(b) if $\ell(t) \in \Sigma$, then t is a leaf in T, and

(c) if $\ell(t) = B \in V$ and t is $n+1$-branching, then there is a rule $B \to x_0 ... x_n \in P$ such that $\ell(tk) = x_k$ for all $k < n+1$.

Corresponds to the fact that c-f grammars cannot rewrite letters anymore.

$B \longrightarrow x_0 \dots x_n$

B

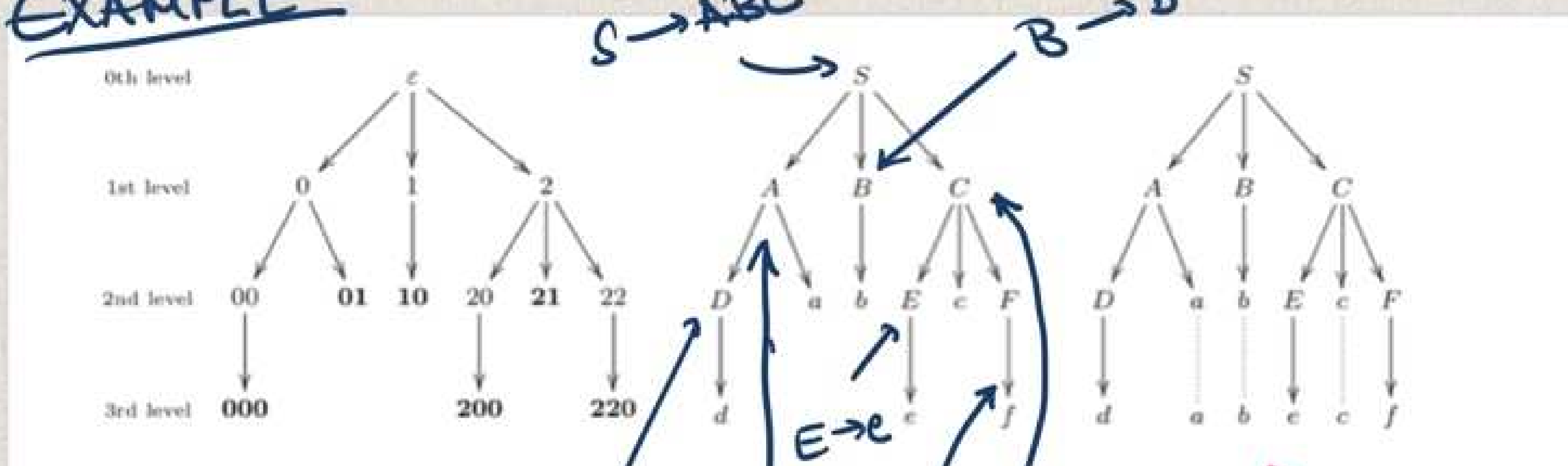
$x_0 \; x_1 \; \dots \; x_n$ $n+1$ succ.

NOTE that this definition allows for variables in terminal nodes of the tree.

EXAMPLE

$S \to ABC$

$B \to b$



$E \to e$

$D \to d$ $\quad$ $A \to Da$ $\quad$ $C \to EcF$

$F \to f$

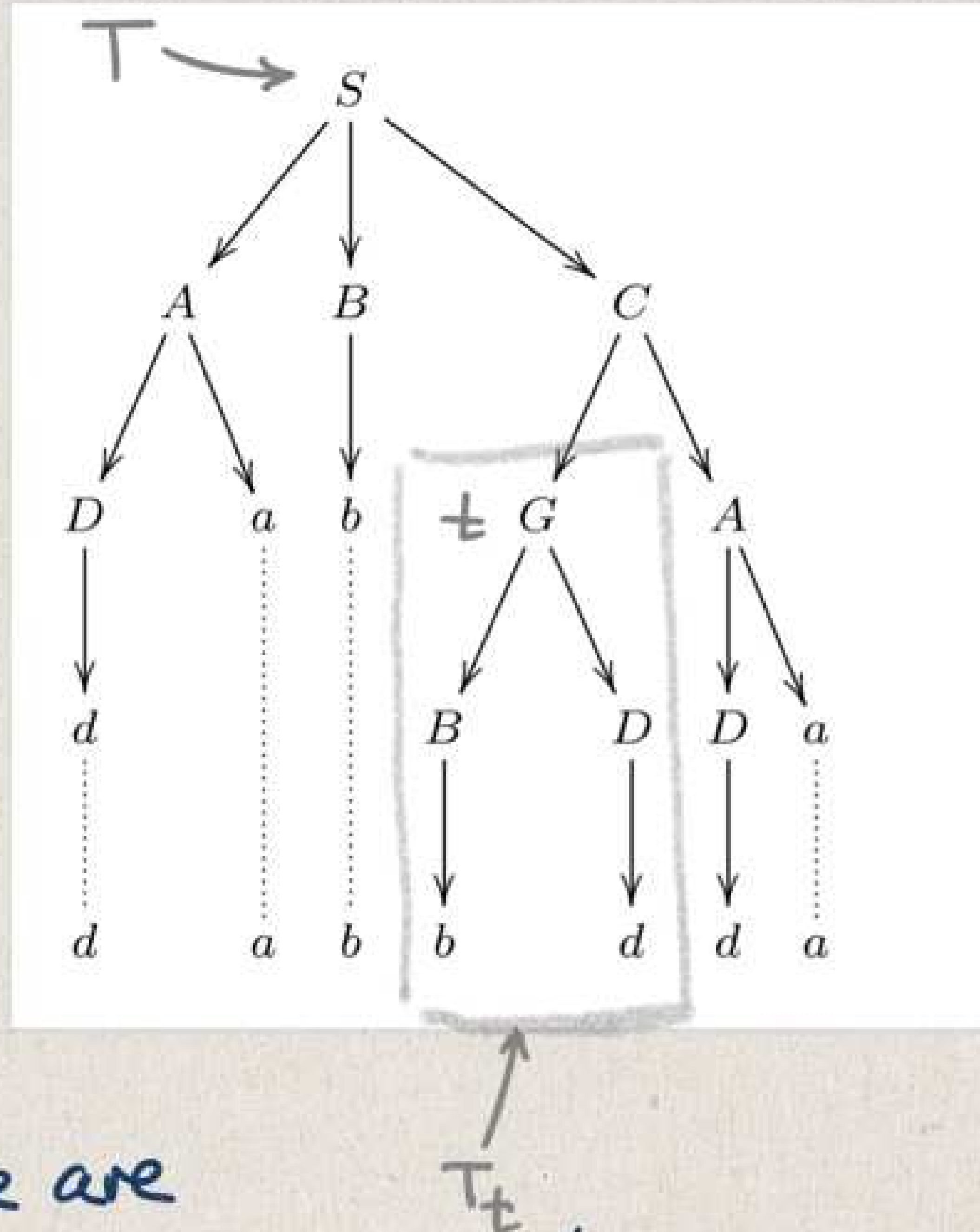
CONVENTION

Draw the leaves of extension to the bottom level and you can read off the word parsed by that tree.

If $\mathbb{T}$ is a parse tree, the sequence of labels of the leaves in left-to-right order is denoted by $\sigma_{\mathbb{T}}$.

Subtrees and substrings

$\sigma_{\pi} = dabbdda$



Then $\pi_t = (T_t, \ell_t)$ with the labelling appropriately defined is a parse tree starting from $\ell(t)$.

$\sigma_{\pi_t} = bd$

By construction, there are τ, τ' s.t. $\sigma_{\pi} = \tau \sigma_{\pi_t} \tau'$.

GRAFTING

Note that if $\mathbf{T}$ and $\mathbf{T}'$ are G-parse trees and $t \in T$ with $\ell(t) = A$ and $\mathbf{T}'$ starts from A, then we can *graft* $\mathbf{T}'$ into $\mathbf{T}$ as follows: we remove $\mathbf{T}_t$ and replace it by $\mathbf{T}'$. By definition, this results in a G-parse tree. More formally, we define $\mathrm{graft}(\mathbf{T}, t, \mathbf{T}') := (S, \ell^*)$ with $S = \{s \in T\,;\, t \not\subseteq s\} \cup \{ts\,;\, s \in T'\}$ and

$$\ell^*(s) := \begin{cases} \ell(s) & \text{if } t \not\subseteq s \text{ and} \\ \ell'(u) & \text{if } s = tu \text{ for some } u \in T'. \end{cases}$$

In terms of the parsed words, grafting a tree $\mathbf{T}'$ into the position of t in $\mathbf{T}$ corresponds to removing the subword $\sigma_{\mathbf{T}_t}$ from $\sigma_{\mathbf{T}}$ and replacing it with $\sigma_{\mathbf{T}'}$. This can be seen in Figure 3.

If $\mathbb{T}$ is a parse tree starting from A, $\ell(t) = B$, $\mathbb{T}'$ is a parse tree starting from B, then $\mathrm{graft}(\mathbb{T}, t, \mathbb{T}')$ is a parse tree starting from A.

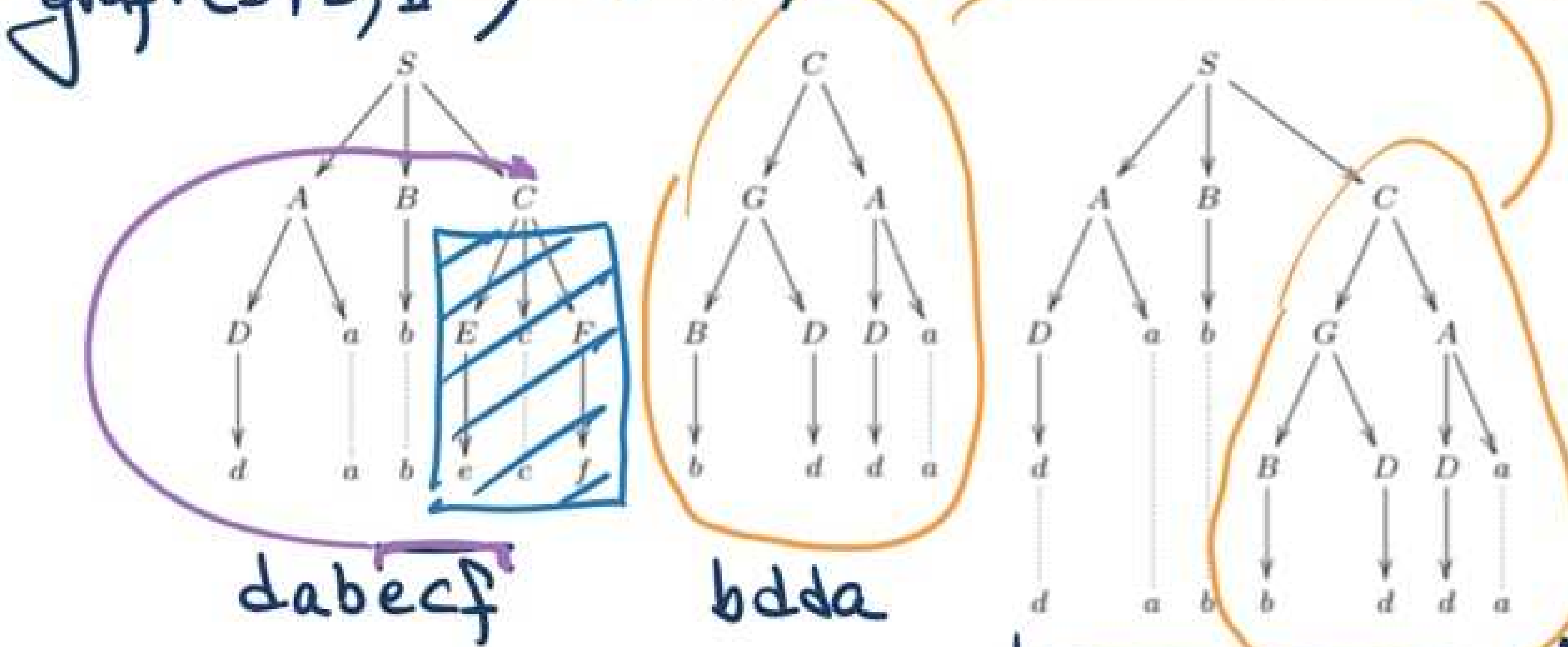


Figure 3: A G-parse tree $\mathbf{T}$ (left), a G-parse tree $\mathbf{T}'$ starting from C, and the result of grafting $\mathbf{T}'$ into the unique node t labelled C in $\mathbf{T}$. Note that $\sigma_{\mathbf{T}} = dabecf$, $\sigma_{\mathbf{T}_t} = ecf$, $\sigma_{\mathbf{T}'} = bdda$, and that $bdda$ replaces ecf in the word parsed by the result of the graft, i.e., $dabbdda$.

dab bdda

If $\sigma_{\mathbb{T}} = \tau \sigma_{\mathbb{T}_t} \tau'$, then

$$\sigma_{\mathrm{graft}(\mathbb{T}, t, \mathbb{T}')} = \tau \cdot \sigma_{\mathbb{T}'} \tau'.$$

So, grafting allows us to take a word in $\mathcal{L}(G)$ and modify it acc. to $\mathbb{T}'$ while staying in $\mathcal{L}(G)$.

PROPOSITION [P 3.2 in the typed notes]

If G is a context-free grammar, then

$w \in \mathcal{L}(G) \iff$ there is a G-parse tree $\mathbb{T}$ starting from S s.t.

$$w = \sigma_{\mathbb{T}}.$$

Proof. Let's call a sequence $\mathbb{T}_0, \dots, \mathbb{T}_n$ of G-parse trees **derivative** if

① $T_0 = \{\varepsilon\}$, $l_0(\varepsilon) = S$

② $\mathbb{T}_{i+1}$ is obtained from $\mathbb{T}_i$ by taking a leaf $t \in T_i$ with $l_i(t) = A \in V$, a rule $A \longrightarrow x_0 \dots x_m \in P$ and adding $m+1$ successor to t, labelled

$$l_{i+1}(tk) := x_k.$$

Obviously, there is a one-to-one correspondence between G-derivations & derivative sequences of G-parse trees. $\sigma_{\mathbb{T}_i}$ is the string derived after i steps of the derivation

This shows "$\Rightarrow$" since if $\mathbb{T}_0, \dots, \mathbb{T}_n$ is such a sequence corresponding to a derivation of w, then $\sigma_{\mathbb{T}_n} = w$.

"$\Leftarrow$". The problem is that the tree $\mathbb{T}$ does not uniquely determine the seq., so we need to pick the derivative sequence on the basis of $\mathbb{T}$.

Let $\mathbb{T}$ be such a parse tree with $\sigma_{\mathbb{T}} = w$.

Fix $\mathbb{T}$ and set $\mathbb{T}_0$ by $T_0 = \{\varepsilon\}$, $\ell_0(\varepsilon) = S$.

In a recursive construction, suppose $\mathbb{T}_i$ is defined already with $T_i \subseteq T$.

If all terminal nodes in T_i are terminal in T, TERMINATE THE CONSTRUCTION.

O/w, there is $t \in T_i$, terminal in T_i, but not terminal in T.

Then add all of the T-successors of t to T_i, producing T_{i+1}; label in the way that $\mathbb{T}_0, \dots, \mathbb{T}_{i+1}$ is a derivative sequence.

Once the construction terminates after step n, i.e., $\mathbb{T}_0, \dots, \mathbb{T}_n$, then $\mathbb{T} = \mathbb{T}_n$, and so $w = \sigma_{\mathbb{T}_n} = \sigma_{\mathbb{T}} \in \mathcal{L}(G)$.

q.e.d.