XI Automata & Formal Languages

Eleventh Lecture

Saturday 29 October 2022

Context-Free Grammars & Parse Trees

Tree $T \subseteq \mathbb{N}^*$ closed under initial segments

for each $t \in T$ there is $n \in \mathbb{N}$ s.t.

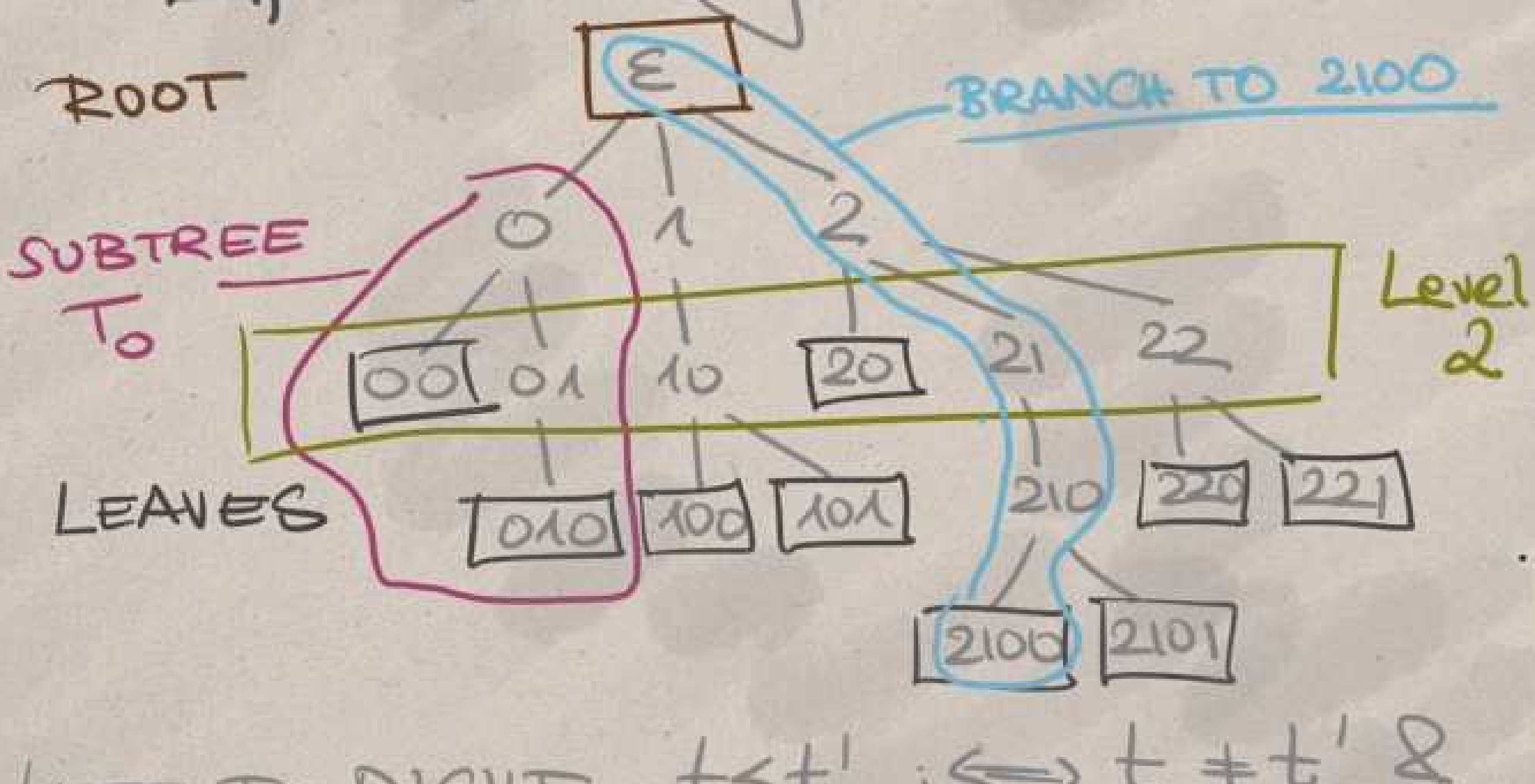
$$tk \in T \iff k < n$$

[t is n-splitting or n-branching]

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

Root : ε

Leaf : 0-branching



LEFT-TO-RIGHT ORDER $t < t' :\iff t \neq t'$ & if k least s.t. $t(k) \neq t'(k)$, then $t(k) < t'(k)$

LEFT-TO-RIGHT ORDER:

- total order on nodes of the same level
- total order on leaves

<u>Parse trees</u> Let G be a c-f grammar.

We call a pair $\mathbb{T} = (T, \ell)$ a <u>G-parse tree</u> if

1. T is a finite tree
2. ℓ is a labelling function with the properties

   2a $\ell(\varepsilon) \in V$ We say $\mathbb{T}$ starts with $\ell(\varepsilon)$.

   2b. If $\ell(t) \in \Sigma$, then t has no successors.

   2c. If t has $n+1$ successors and $\ell(t) = A \in V$ [by 2b]

   then there is a rule
   $$A \longrightarrow x_0 \dots x_n \in P$$
   s.t. for all $k \leq n$
   $$\ell(tk) = x_k$$

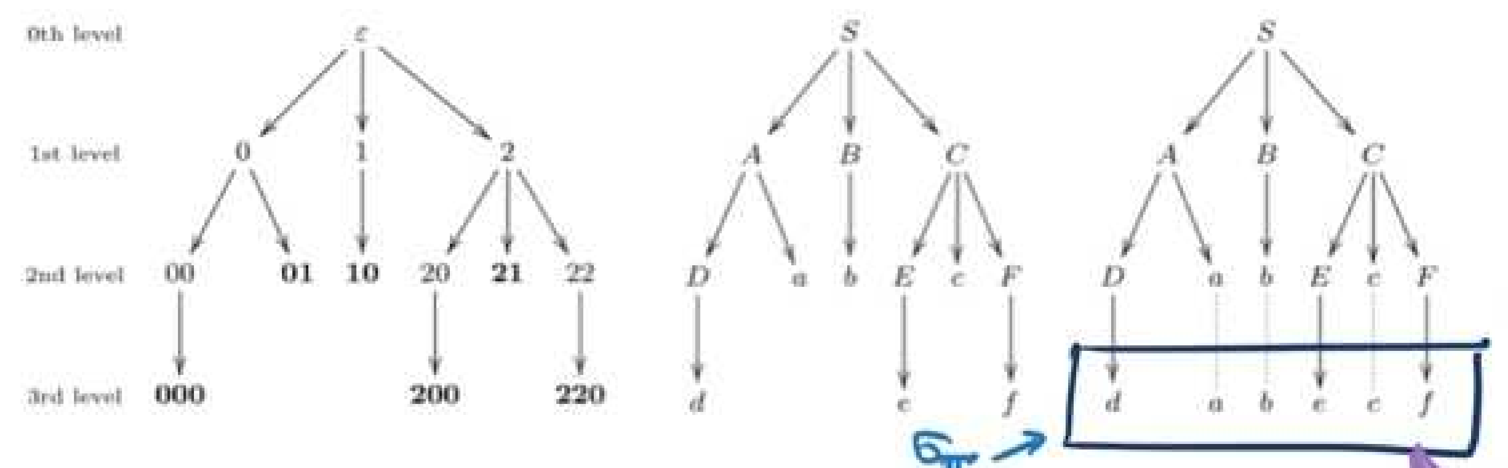


Figure 2: *Left.* A finitely branching tree; leaves are marked in boldface font. *Middle.* The same tree labelled to form a G-parse tree. *Right.* The same parse tree with the leaves extended to the final level to highlight that the parse tree parses the word $dabecf$.

If $\mathbb{T} = (T, \ell)$ is a G-parse tree, and $t_0, \dots, t_m$ are its leaves written in the left-to-right order:

$$t_0 < t_1 < \dots < t_m$$

then the string parsed by $\mathbb{T}$ is

$$\sigma_{\mathbb{T}} = \ell(t_0) \dots \ell(t_m)$$

$\sigma_{\mathbb{T}}$ can be easily read off the labelled tree by extending all leaves to the height of T and reading from left to right.

Note that if $t \in T$, then

$$\sigma_{\mathbb{T}} = \alpha \, \sigma_{\mathbb{T}_t} \beta$$

for some α, β. where

$$\mathbb{T}_t = (T_t, \ell_t) \text{ and } \ell_t(s) := \ell(ts)$$

Proposition 3.2 Let G be context-free.

Then for every w:

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

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

Proof. First we observe that certain sequences of parse trees correspond to derivations.

A sequence $\mathbb{T}_0, \dots \mathbb{T}_n$ of G-parse tree is called DERIVATIVE if

$\mathbb{T}_0 = (\{\varepsilon\}, l_0)$ with $l_0(\varepsilon) = S$.

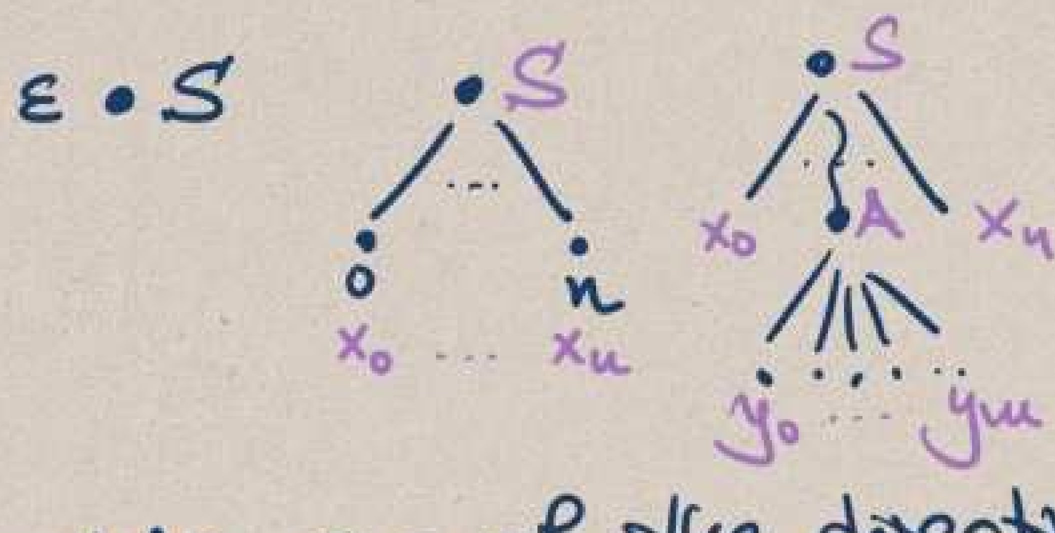
and for all i, $T_{i+1} \supseteq T_i$ constructed as follows:

there is a leaf t in T_i with $l_i(t) = A \in V$ and $A \to x_0 \dots x_n \in P$ and in T_{i+1}, t has $n+1$ successors

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

There is a 1-to-1 correspondence between G-derivations starting from S and derivational sequences of parse trees.

$$
\begin{array}{ccccccc}
S=\sigma_0 & \xrightarrow[\ 1\ ]{G} & \sigma_1 & \xrightarrow[\ 1\ ]{G} & \cdots & \xrightarrow[\ 1\ ]{G} & \sigma_n \\
| & S\to x_0\cdots x_n & & A\to y_0\cdots y_m & & & \\
\mathbb{T}_0 & & \mathbb{T}_1 & & & & \mathbb{T}_n .
\end{array}
$$



This yields one of the directions of our Proposition 3.2:

If $S \xrightarrow{G} w$, then $\mathbb{T}_n$ in the above 1-to-1 correspondence is a G-parse tree with $\sigma_{\mathbb{T}_n} = w$.

For the converse, assume $\mathbb{T}$ is a G-parse tree with $\sigma_{\mathbb{T}} = w$.

Construct derivative sequence of parse trees

Start from $\mathbb{T}_0 = (\{\varepsilon\}, \ell\restriction\{\varepsilon\})$.

In each step, assume $\mathbb{T}_0, \dots, \mathbb{T}_i$ is already a derivative sequence with $T_i \neq T$. Say $t \in T \setminus T_i$.

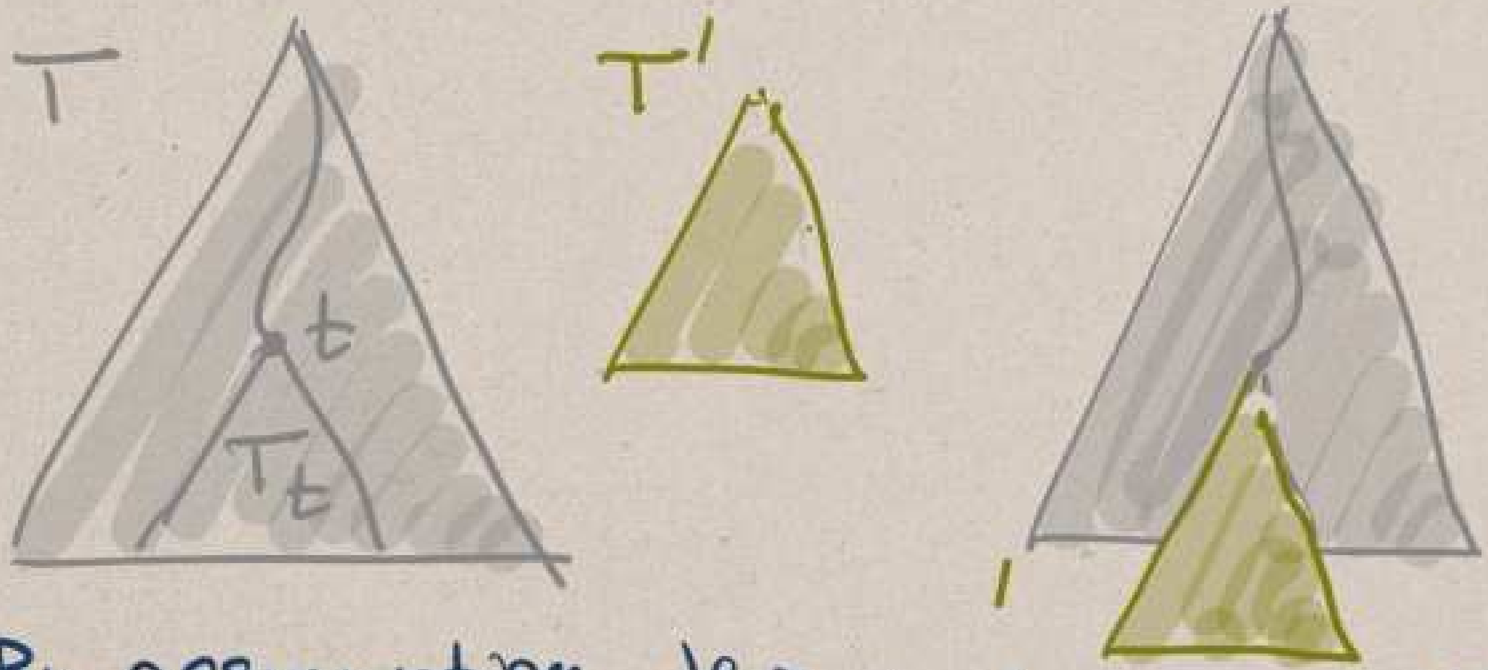
So, there is a terminal node in T_i on the branch to t which is not terminal in T. Create T_{i+1} by adding the T-successors of t to T_i.

Since T is finite, after finitely many steps, have $\mathbb{T} = \mathbb{T}_m$.

So $\mathbb{T}_0, \ldots, \mathbb{T}_m$ is a derivative seq, and thus $S \xrightarrow{G} \sigma_{\mathbb{T}_m} = \sigma_{\mathbb{T}} = w$.

q.e.d.

<u>Grafting</u> $\mathbb{T}$ parse tree
$t \in T$ $\ell(t) = A$
$\mathbb{T}'$ parse tree starting from A



By assumption, this produces a parse tree.

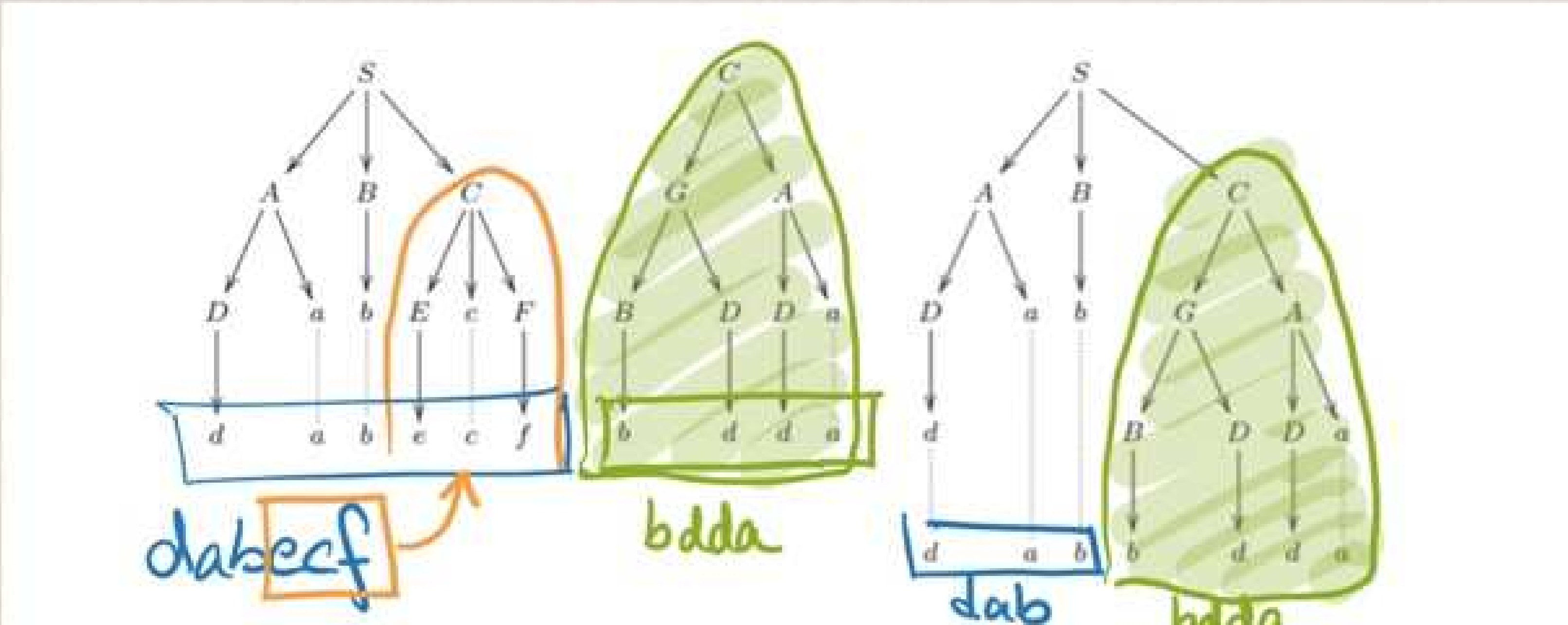


Figure 3: A G-parse tree $\mathbf{T}$ (left), a G-parse tree $\mathbf{T}'$ starting from C, and the result of grafting $\mathbf{T}'$ into the unique node t labelled C in $\mathbf{T}$. Note that $w_{\mathbf{T}} = dabecf$, $w_{\mathbf{T}_t} = ecf$, $w_{\mathbf{T}'} = bdda$ and that $bdda$ replaces ecf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

$$\mathrm{graft}(\mathbb{T}, t, \mathbb{T}') := (S, \ell^*)$$

where

$$S := \{ s \in T ;\ t \not\preceq s \} \cup \{ tv ;\ v \in T' \}$$

$$\ell^*(s) := \begin{cases} \ell(s) & t \not\preceq s \\ \ell'(v) & \text{if } s = tv \text{ for some } v \in T'. \end{cases}$$

Obviously

$$\sigma_{\mathrm{graft}(\mathbb{T}, t, \mathbb{T}')} = \alpha \sigma_{\mathbb{T}'} \beta$$

where $\sigma_{\mathbb{T}} = \alpha \sigma_{\mathbb{T}_t} \beta$.

§3.2 CHOMSKY NORMAL FORM (CNF)

A grammar is in Chomsky Normal Form if all of its rules are of the form

BINARY (a) $A \longrightarrow BC$ $\quad A, B, C \in V$

UNARY (b) $A \longrightarrow a$ $\quad A \in V, a \in \Sigma$

Clearly, every CNF grammar is context-free, but "most" c-f grammars are not in CNF.

<u>Lemma 3.3</u> If G is a grammar in CNF and $w \in \mathcal{L}(G)$ with $|w| = n$, then every G-derivation of w has length $2n-1$.

<u>Proof.</u> Binary rules increase length by one. Unary rules preserve length.
So there are precisely $n-1$ binary rule applications.

So, in order to have 0 variables and n letters in w,

many rules decrease # var. by 1
increase # letter by 1,

so there must be exactly n many rule applications.

Together $n-1+n = 2n-1$.

q.e.d.

GOAL Theorem by Chomsky:

Every context free grammar G has a CNF grammar G' s.t.

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

Moreover, there is an algorithm to produce G' from G.

Three problems for the proof:

(1) rules of the form

$A \longrightarrow X_0 \dots X_n$ where $n \geq 2$

MIXED RULES (2) rules of the form

$A \longrightarrow \alpha$ where α mixes letter and variables

UNIT RULES (3) rules of the form

$A \longrightarrow B$

[rules of the form $A \longrightarrow ab$ for $a, b \in \Sigma$ will be handled with this in one go]