CSE396 Lecture Thu 3/30 5pr 2017

Suppose we have a "Target Specification" T of a language: T defined by prose, sets, or machine, etc.
Suppose we have a CFG G that is trying to "model" T.

G is <u>sound</u> for the spec if $L(G) \subseteq T$. G has no false positives

G is <u>comprehensive</u> for it if $T \subseteq L(G)$. "no false negatives"

These concepts were first formalized in logic where G is generalized to a "Formal System F" — kind of like a grammar where a string is generated by 2 others not just 1.
T = the set of <u>true</u> statements. <u>Sound</u> means $L(F) \subseteq T$,
L(F) = the set of <u>theorems</u> of F. i.e. "every theorem proved is true."

Comprehensiveness would mean $T \subseteq L(F)$, i.e. that F could prove every true statement (over a particular logical "alphabet")
But, Kurt Gödel proved that no sound and executable formal system can be comprehensive for T = { true arithmetical stmts }
I.e. for any sound and effective F over the "alphabet of arithmetic"
$L(F) \subsetneq T$. Gödel's Incompleteness Theorem
Uncomprehensiveness

We will think of the concepts most with CFGs and apply them even when T is given by another grammar.

If we change an original grammar G into G_2, then

- the change is sound if $L(G_2) \subseteq L(G)$.
- But of course we want $L(G_2) = L(G)$: comprehensiveness too.

<u>Def</u>: A CFG G_2 is in <u>Chomsky Normal Form (ChNF)</u> if every rule $A \to \vec{X}$ either has $\vec{X} \in \Sigma$ or $\vec{X} \in V \cdot V$.

ie has the form $A \to c$ or $A \to BC$, B, C possibly $= A$.

Our text enables "ChNF" grammars to generate ε by the special exception that we can add an extra start symbol S_0 and rules $S_0 \to \varepsilon \mid \ldots$ all right hand sides of S.

<u>Def^n</u>: A variable $A \in V$ is <u>nullable</u> if $A \Rightarrow^* \varepsilon$.

Note $\varepsilon \in L(G) \Leftrightarrow S$ is nullable.

<u>Theorem</u>: Given any CFG G, we can build a CFG G_4 in ChNF s.t.

$$L(G_4) = L(G) \setminus \{\varepsilon\} \quad \text{if we regard ChNF strictly}$$
$$= L(G) \quad \text{if } \varepsilon \in L(G) \text{ and we allow the "}S_0\text{" fudging above.}$$

<u>Step 1</u> will build G_1 st. $L(G_1) = L(G) \setminus \{\varepsilon\}$ and G_1 has no nullable variables.

Algorithm and Proof of Step 1: Given $G = [V, \Sigma, R, S]$

i• First identify the subset NULLABLE $\subseteq V$ of nullable variables.

ii• For every rule $A \to \vec{X}$ where $\vec{X}$ has nullable variables, add the rules $A \to \vec{X}'$ for all combinations of deleting one or more occurrences of the nullable variables in $\vec{X}$

Sound.

iii• Delete all ε-rules $B \to \varepsilon$, incl. any new ones.

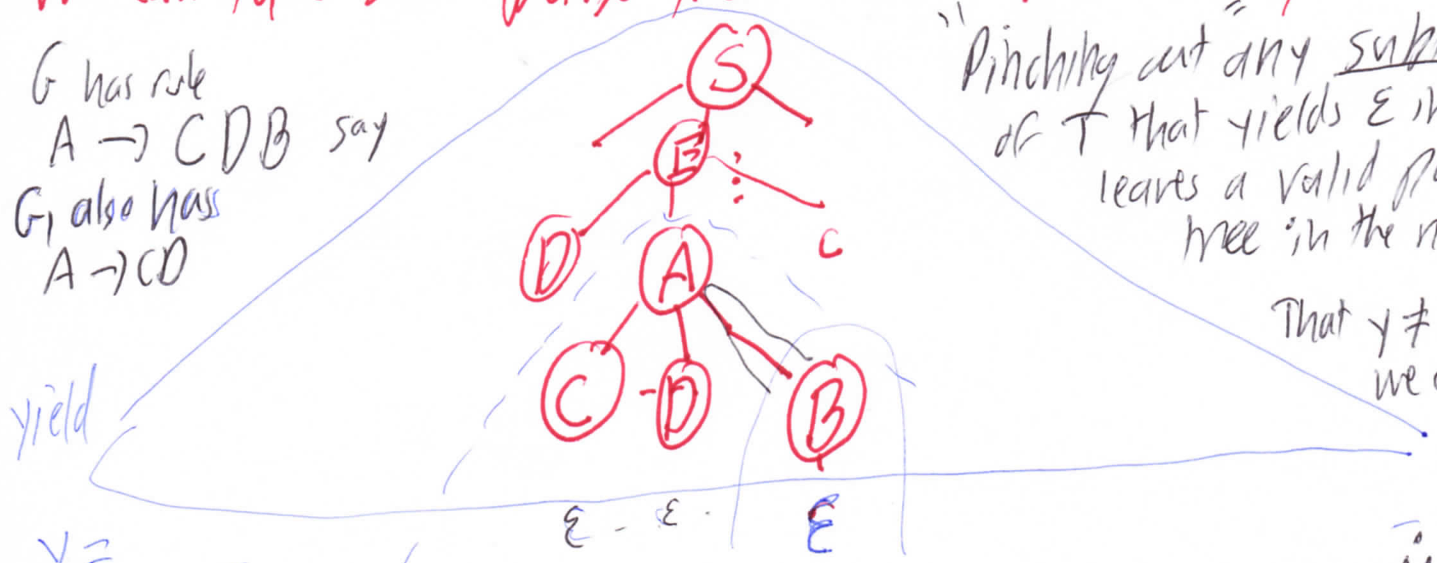
We will show this comprehensive except for ε itself.

Proof of substep [iii]: Let any $y \neq \varepsilon$ in $L(G)$ be given. Then we can take some parse tree T for y in the original G.

G has rule $A \to CDB$ say

G_1 also has $A \to CD$

"Pinching out" any subtree of T that yields ε inside y leaves a valid parse tree in the new G_1



That $y \neq \varepsilon$ means we don't pinch out all of T.

If $A \Rightarrow^* \varepsilon$ inside T, then G_1 has the rule $E \to Dc$ as well as $E \to DAc$.

$\therefore y \in L(G_1)$ so OK. ☒

Algorithm for telling which vars are NULLABLE.

1. Initialize $NULL = \{A \in V : A \to \varepsilon \text{ is a rule}\}$.
2. bool changed = true
3. while (changed) {
    changed = false;
    for (each rule $A \to \vec{X}$ in R) {
        if ($\vec{X} \in (NULL)^*$ and $A \notin NULL$) {
            $NULL = NULL \cup \{A\}$
            changed = true;
        }
    }
   }
4. Output final NULL.

— Halts within $|V|$ iterations because each iteration either enlarges NULL or leaves the flag <u>changed</u> as false.

Target: NULL = NULLABLE
<u>Sound</u> because if <u>we</u> add A to NULL, then $A \Rightarrow \vec{X} \in NULL^*$
<u>Comprehensive</u> → "think about it"

<u>Two Examples.</u>

$G: S \to AB \mid cA \mid \varepsilon$
$A \to \cancel{\varepsilon}\ SS \mid Sa$
$B \to AS \mid c$

with $A \to \varepsilon$ → $NULL = \{S, A\}$
$B \to AS$ puts B into NULL at the first iteration

with $A \to SS$
$NULL = \{S\}$.
$A \to SS$ puts A into NULL
$B \to AS$ puts B into NULL.

$S \to \varepsilon \mid (S) \mid SS$ NULLABLE $= \{S\}$

$G_1: S \to () \mid (S) \mid SS$

G_1 generates all <u>nonempty</u> balanced-() strings.