CSE396 Lecture Tue 3/29 Spring 2016

Recall

G_2: $S \to \varepsilon \mid AB \mid BA \mid SS$ $L = \{x : \#a(x) = \#b(x)\}$

$A \to a \mid aS \mid BAA$. $L_A = \{x : \#a(x) = \#b(x) + 1\}$

$B \to b \mid bS \mid ABB$ $L_B = \{x : \#a(x) = \#b(x) - 1\}$.

For $\underline{L(G_2) \subseteq L}$ (soundness) define P_S, P_A, P_B, where for instance

P_A = "Every string x that I derive has $\#a(x) = \#b(x) + 1$."

For $\underline{L \subseteq L(G)}$ (comprehensiveness) we do the opposite: $P'(n) = P(n) \wedge Q(n) \wedge R(n)$

where $Q(n) \equiv$ For each $x \in \Sigma^n$, if $x \in L_A$ then $A \Rightarrow^* x$, which kind-of says: "I derive every string x that has $\#a(x) = \#b(x) + 1$."

Proving Comprehensiveness $\equiv$ Proving an algorithm for $\underline{\text{parsing}}$.

E.g. Given $x = babaaab$, how does x get derived? Is $x \in L_S, L_A, L_B$, or "none of the above"?

$S \to \varepsilon \mid AB \mid BA \mid SS$ ← unnecessary

$A \to a \mid AS \mid BAA$

$B \to b \mid BS \mid ABB$

$\underline{x \in L_A}$. Proof does subcases for the first letter in x.

$\underline{x = ay:}$ $\Longrightarrow y \in L_S$ so do $A \Rightarrow \boxed{AS \Rightarrow aS} \Rightarrow^* ay = x$.

$\underline{x = bz:}$ Hence we can parse $x =: buv$ where $|bu| = j$, i.e. $\text{diff}(bu, j) = 0$

Proof notes that the Diff function $\text{diff}(x, i) = \#a(x_1 \cdots x_i) - \#b(x_1 \cdots x_i)$

- starts at 0 $\boxed{\text{steps } \pm 1 \text{ at each char}}$
- Goes to $\boxed{-1}$ since x begins with b
- Ends at $\boxed{+1}$ since $\#a(x) = \#b(x) + 1$

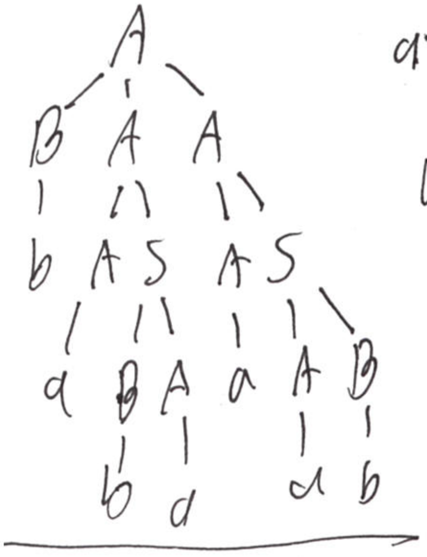
∴ By a "Discrete" version of the Intermediate Value Theorem, $\exists i:$ $\text{diff}(x_1 \cdots x_i) = 0$.

b	$u = a$ or aba	$V = ba^3b$ or aab

Diff 0 −1 0 +1

$x = b \cdot aba \cdot aab$ $A \Rightarrow BAA \Rightarrow bAA$
$\Rightarrow bAS \Rightarrow ba$

$$A \Rightarrow BAA \Rightarrow b\underbrace{\overset{aab}{AA}}_{aba} \Rightarrow bASA \Rightarrow baSA \Rightarrow baBAA \Rightarrow babAA \Rightarrow babaA \Rightarrow babaaS \Rightarrow babaaAB \Rightarrow^2 babaaab.$$

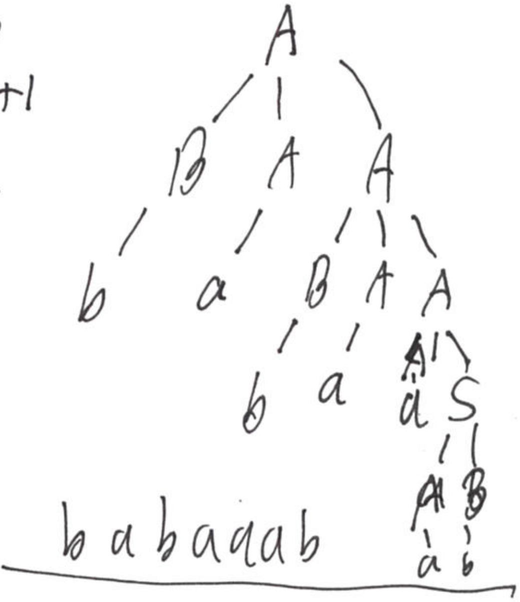


babaaab.

Using the "least j" rule ~~breaks~~ parses babaaab as

$$b \cdot a \cdot b\,\underbrace{aaab}_{1}$$

Diff 0 -1 0 +1



babaaab

Refined grammar

$S \to \varepsilon \mid aB \mid bA$

$A \to aS \mid b\underline{AA}$

By symmetry, can use $B \to bS \mid a\underline{BB}$

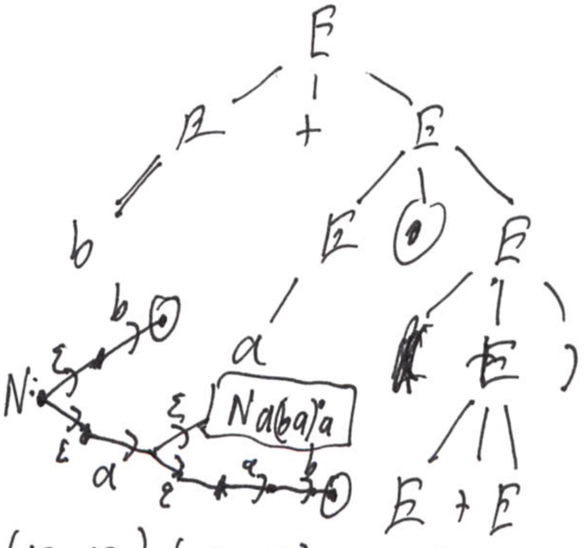
This grammar G_2' remains comprehensive: $L \subseteq L(G_2')$ by what the proof actually used.

Unfortunately G_2' is still ambiguous, as the case $x = babaaab$ showed.

Regular Expression Parsing:

$$b \cup a(a(ba)^* a \cup ab)$$

$$\underbrace{b}_{}\overset{\uparrow}{+}\underbrace{a\cdot(a\cdot(b\cdot a)^*\cdot a \overset{\uparrow}{+} a\cdot b)}_{}$$



Fully parenthesized: $E \to \underset{\sim}{\emptyset} \mid \underset{\sim}{\varepsilon} \mid \underset{\sim}{a} \mid \underset{\sim}{b} \mid (E+E) \mid (E\cdot E) \mid (E^*)$

Thus the Regexp-to-NFA proof became was one by Structural Induction.

Chomsky Normal Form: $\longrightarrow$ Makes the proof of the CFL Pumping lemma in §2.3 nicer.

<u>Def'n</u>: A CFG $G = (V, \Sigma, R, S)$ is in Chomsky Normal Form (ChNF) if every rule $A \to X$ has $X \in \Sigma$ or $|X| = 2$ with $X \in VV$.

Allowed: $A \to c$ ($c \in \Sigma$), $A \to AA$, $A \to BC$

Not Allowed: $\underbrace{A \to \varepsilon}_{\varepsilon\text{-rule.}}$ $A \to \underbrace{aS}_{\text{mixed RHS}} \mid \underbrace{A \to B}_{\text{unit rule}}$ $\underbrace{A \to BCD}_{|RHS| \geq 3}$

(Text allows $\underline{S \to \varepsilon}$ provided S is not on the RHS of any rule.) Can always put ε in last by a new start var. S_0 and rules $S_0 \to \varepsilon \mid \left(\begin{smallmatrix}\text{RHS}\\ \text{of } S\end{smallmatrix}\right)$.

<u>Theorem</u>: Given any CFG G, we can build G' in ChNF st. $L(G') = L(G)$.

Proof. By <u>Algorithm</u> ① Identify the set NULLABLE of variables A st. $A \Rightarrow^* \varepsilon$.

(not the fastest possible) ② For every rule $A \to X$ include every rule $A \to X'$

...r blends ① & ② obtained by erasing 1 or more <u>occurrences</u> of vars in NULLABLE.

Idea: Suppose $A \to X$ is $A \to \underline{B}\underline{C}DB$ and B, C are nullable.

(Keep ...$\Rightarrow BCDB$) Derivations can do $B \Rightarrow^* \varepsilon$ and $C \Rightarrow^* \varepsilon$, hence they can do all of:

...$\Rightarrow BD$, $\underline{A \Rightarrow^* D}$, $A \Rightarrow^* DB$, $A \Rightarrow^* BCD$, $A \Rightarrow^* CD$, $A \Rightarrow^* CDB$, $A \Rightarrow^* BDB$

③ Then delete all ε-rules. Got G_1 st. $L(G_1) = L(G) \setminus \{\varepsilon\}$.

(...volves a ...ansitive ...osure of ...$\Rightarrow B$ relation.) ④ Determine which $A, B \in V$ allow $A \underset{G_1}{\Rightarrow^*} B$ (note: we may get <u>more</u> unit rules from step 2)

⑤ For each such A, B $(B \neq A)$, make every RHS of B a RHS of A.

⑥ Then <u>delete</u> all unit rules to get G_2 st. $L(G_2) = L(G) \setminus \{\varepsilon\}$.

Alias every terminal a to a variable X_a, and ⑧ Use dummy variables to reduce RHS to 2.

Example: $A \to BCdB$, then add variables X_d, Y_1, Y_2 and do:

$A \to BY_1, \quad Y_1 \to CY_2, \quad Y_2 \to X_d B, \quad X_d \to d.$

<u>Added</u> — for Thursday — Full Examples:

$S \to \varepsilon \mid (S)S \qquad L = \{x \in \{(,)\}^* : x \text{ is } \underline{\text{balanced}}\}$

1. NULLABLE $= \{S\}$ since $S \Rightarrow^* \varepsilon$

2. Add rules: $S \to () \mid (S) \mid ()S$ $\quad L(G_1) = L \setminus \{\varepsilon\}$.

3. Delete $S \to \varepsilon$: $G_{\underline{1}} = S \to (S)S \mid () \mid (S) \mid ()S$

4. No unit rules were introduced or were there previously. 5.6. <u>can skip</u>.

7. Alias $L \to$ '(' $R \to$ ')'. $G_2 = S \to LSRS \mid LR \mid LSR \mid LRS \mid$ $L \to ($ $R \to)$

8. Add variables Y_1, Y_2, Y_3, Y_4.

$$\boxed{\begin{array}{l} G_3 = S \to LY_1 \mid LR \mid LY_3 \mid LY_4, \; L \to (, R \to) \\ \quad Y_1 \to SY_2, \; Y_2 \to RS, \; Y_3 \to SR, \; Y_4 \to RS \end{array}}$$

Yes we could economize with $Y_4 \equiv Y_2$, but who cares? G_3 is UGLY!!!

9. Now $L(G_3) = L \setminus \{\varepsilon\}$. If we want to put ε back in the language, do:

$G' = S_0 \to \varepsilon \mid LY_1 \mid LR \mid LY_3 \mid LY_4 \quad Y_1 \to SY_2, \; Y_2 \to RS, \; Y_3 \to SR, \; Y_4 \to RS$

$\quad\; S \to LY_1 \mid LR \mid LY_3 \mid LY_4 \quad L \to$ '(', $R \to$ ')'! $\boxed{\text{OK in text ChNF}}$

We couldn't add $S_0 \to S$ since that would be a unit rule. Then $\underline{L(G') = L(G) = L}$

The algorithm to define NULLABLE is an important kind of iterative loop akin to Breadth-First Search. Note that if we initialize NULLABLE $= \emptyset$ instead, then since $\emptyset^* = \underline{\varepsilon}$, we still get the ε-rule variables on the first iteration.

```
NULLABLE := { A ∈ V : A → ε is a rule };
bool foundNew = true;
while (foundNew) {
   foundNew = false;
   foreach (rule A → X in R) {
      if (X ∈ NULLABLE*)  {
         NULLABLE = NULLABLE ∪ {A};
         foundNew = true;
} } }
```