Exercise (7.3.1. from textbook) 14/12/2005

Show that the context-free languages are closed under operation init, defined as follows:

for a language L over Σ:

$$init(L) = \{ w \mid w \cdot x \in L, \text{ for some } x \in \Sigma^* \}$$

Proof:

Consider a CFL L and a CFG $G = (V_N, V_T, P, S)$ in Chomsky normal form without useless symbols s.t. $\mathcal{L}(G) = L$.

We want to construct a CFG G^{init} for $init(L)$.

For a non-terminal A, let $G_A = (V_T, V_N, P, A)$ the grammar identical to G, but with A as start-symbol. Let $L_A = \mathcal{L}(G_A)$.

Idea: for each NT A, we introduce a new NT $\overline{A}$ that generates $init(L_A)$.

We construct $G^{init} = (V_N^{init}, V_T, P^{init}, S^{init})$ with

$V_N^{init} = V_N \cup \{ \overline{A} \mid A \in V_N \}$

$S^{init} = \overline{S}$

and P^{init} defined as follows:

- for every production $A \to BC$ in P, we have in P^{init}:

  $A \to BC \qquad \overline{A} \to B\overline{C} \mid \overline{B}$

- for every production $A \to a$ in P, we have in P^{init}:

  $A \to a \qquad \overline{A} \to a \mid \varepsilon$

We prove that $\mathcal{L}(G^{init}) = init(L)$

"$\subseteq$" Let $w \in \mathcal{L}(G^{init}_{\bar{A}})$. We show that $w \in init(L_A)$, i.e. there is $x \in \Sigma^*$ s.t. $w \cdot x \in L_A = \mathcal{L}(G_A)$.

We proceed by induction on the depth n of the derivation tree for w.

base case ($n=1$)

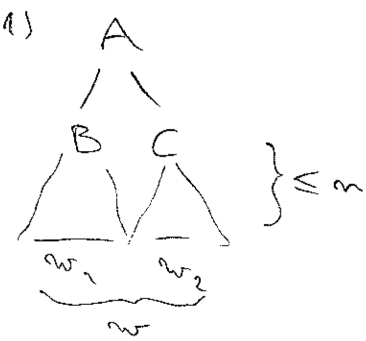
case 1) $\begin{array}{c}\bar{A}\\ |\\ a\end{array}$ and $a \in L_A$

case 2) $\begin{array}{c}\bar{A}\\ |\\ \varepsilon\end{array}$ and $\varepsilon \in init(L_A)$

inductive case: assume that, if there is a derivation tree of depth $\leq n$ showing $w \in \mathcal{L}(G_A^{init})$, we have that $w \in init(L_A)$.

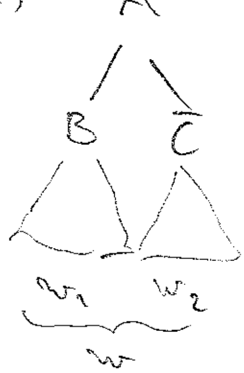
Consider a derivation tree of depth $n+1$:

case 1)



$w \in L_A \subseteq init(L_A)$

case 2)



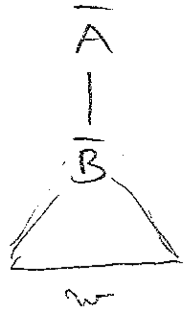
$w = w_1 \cdot w_2$

by IH, $w_2 \in init(L_C)$, i.e. there is $x \in \Sigma^*$ s.t. $w_2 \cdot x \in L_C$, i.e. $C \Rightarrow^* w_2 x$.

Since $A \to B\bar{C}$ is in P^{init}, we have that $A \to BC$ is in P.

Since $B \Rightarrow^* w_1$, we have that $A \Rightarrow^* w_1 w_2 \cdot x$.

Hence $w \in init(L_A)$

case 3) 

by I.H., $w \in init(L_B)$, i.e.
there is $x \in \Sigma^*$ s.t. $w \cdot x \in L_B$,
i.e. $B \Rightarrow^* w \cdot x$.

Since $\bar{A} \to \bar{B}$ is in P^{init}; there is some C s.t. $A \to BC$ is in P.

Since G contains no useless symbols, C is generating, i.e. $C \Rightarrow^* y$ for some y.

Hence $A \Rightarrow B \cdot C \Rightarrow^* w \cdot x \cdot y$, and $w \in init(L_A)$.

"$\supseteq$" Let $w \in init(L)$. Then there is $x \in \Sigma^*$ s.t. $w \cdot x \in \mathcal{L}(G)$.

E. 9.1

Exercise (7.3.1 from textbook) 7/12/2005

Show that the context-free languages are closed under operation init, defined as follows: given a language L,

$$\text{init}(L) = \{ w \mid \text{for some } x,\ wx \in L \}$$

solution

Consider a CFL L, and a grammar G for L.
For each nonterminal A in G, we want to have an additional nonterminal $\bar{A}$ that generates init(L_A), where L_A is the language generated by G having A as axiom (start symbol).
The goal is to construct a grammar (a CFG) that generates init(L). The start symbol of $\bar{G}$ is $\bar{S}$.
Without loss of generality, we can assume that G is in Chomsky normal form.
For every production $A \to BC$ in G, in $\bar{G}$ we have:

$$\bar{A} \to B\bar{C} \mid \bar{B}$$
$$A \to BC$$

and for every production $A \to a$ in G, in $\bar{G}$ we have

$$A \to a$$
$$\bar{A} \to \epsilon.$$

The obtained grammar (which is not in Chomsky normal form) generates init(L) — formal proof left to the reader.

E 9,2

Exercise (7.3.3 from textbook) 7/12/2005

Show that CNF languages are not close under operator min, defined as follows:

given a language L,

$$\min(L) = \{ w \mid w \in L \text{ but no proper prefix of } w \text{ is in } L \}$$

solution

The proof goes through a counterexemple that shows that the closure does not hold.

Consider the CFL

$$L = \{ a^i b^j c^k \mid k \geq i \} \cup \{ a^i b^j c^k \mid k \geqslant j \}$$

L is clearly a CFL, since it is the union of two language for which we can straightforwardly write a CFG.

Now, notice that

$$\min(L) = \{ a^i b^j c^k \mid k = \min(i,j) \}$$

This language is not a CFL, and this can be proved by using the pumping lemma. Let n be the pumping lemma constant; consider $w = a^n b^n c^n \in \min(L)$.

Exercise (8.1.1 from textbook) 14/12/2005

E 9.3

Give a reduction from the hello-world problem to the following problem:

given a program P and an input I, does P eventually halt when it is given I as input?

solution

We take P and modify it in the following way:

(1) We make sure it never halts unless we explicitly want it. This is done by inserting some instruction like

```
while (1) {};
```

at the end of main() and where there is a return in main().

(2) We make P record the first 12 characters printed; if they are "hello, world" the program halts.

In this way the modified program halts if and only if the original program P prints "hello, world". This ends our reduction: in fact, if we are able to decide whether a program eventually halts, then we are able to decide whether it prints "hello, world".

E 9.4

Exercise (Example 8.2 from textbook) 14/12/2005

Construct a Turing Machine accepting the language

$$\{0^n 1^n \mid n \geq 1\}$$

solution

The idea is that the TM M that we construct reads the leftmost 0, turns it into X, and moves right until it reaches a 1, that is turned into Y. Then the head moves left again to the leftmost 0 (on the right to a X), and starts again until all 0's and 1's are turned into X's and Y's respectively.
If the input is not in $0^* 1^*$, M will fail to find a move and it won't accept. If M changes the last 0 and the last 1 in the same round, it will go into the final state and accept.

$$Q = \{q_0, q_1, q_2, q_3, q_4\}$$
$$\Sigma = \{0, 1\}$$
$$\Gamma = \{0, 1, X, Y, b\} \qquad (b \text{ denotes blank symbol})$$

q_0 : start state

$$F = \{q_4\}$$

In q_0 is the state in which M is when the head preceeds the leftmost 0. In state q_1, M moves right skipping 0's and Y's until it gets to a 1. In state q_2, M moves left while skipping Y's and 0's again, until it gets to a X and goes again in q_0.

(E 9,5)

Starting from q_0, if a Y is read instead of a 0, M gets in q_3 and moves right: if a 1 is found, then there are more 1's than 0's; if a b is read, then the initial string is accepted (transition to q_4).

	0	1	X	Y	b
q_0	(q_1, X, R)	—	(q_3, Y, R)	—	—
q_1	$(q_1, 0, R)$	(q_2, Y, L)	—	(q_1, Y, R)	—
q_2	$(q_2, 0, L)$	—	(q_0, X, R)	(q_2, Y, L)	—
q_3	—	—	—	(q_3, Y, R)	(q_4, b, R)
q_4	—	—	—	—	—

Exercise

Show the computation of the TM above when the input string is:

(a) 00

(b) 000111

solution

(a) $q_0 00 \vdash X q_1 0 \vdash X 0 q_1$

and the TM halts

(b) $q_0 000111 \vdash X q_1 00111 \vdash X 0 q_1 0111 \vdash$
$X00 q_1 111 \vdash X0 q_2 0Y11 \vdash X q_2 00Y11 \vdash q_2 X00Y11 \vdash$
$X q_0 00Y11 \vdash XX q_1 0Y11 \vdash XX0 q_1 Y11 \vdash XX0Y q_1 11 \vdash$
$XX0 q_2 YY1 \vdash XX q_2 0YY1 \vdash X q_2 X0YY1 \vdash XX q_0 0YY1 \vdash$
$XXX q_1 YY1 \vdash XXXY q_1 Y1 \vdash XXXYY q_1 1 \vdash XXXY q_2 YY \vdash$
$XXX q_2 YYY \vdash XX q_2 XYYY \vdash XXX q_0 YYY \vdash XXXY q_3 YY \vdash$
$XXXYY q_3 Y \vdash XXXYYY q_3 b \vdash XXXYYYb q_4 b$

E 9.6

Exercise (8.22 from textbook) :

Design Turing machines accepting the following languages :

$\{ w \in \{0,1\}^* \mid w$ has an equal number of 0's and 1's $\}$

Solution

The idea is that the head of our TM M moves back and forth on the tape, "deleting" one 0 for each 1 ; if there are no 0's and 1's in the end, the string is accepted.
When in state q_1, M has found a 1 and looks for a 0 ; in state q_2 is the other way around.
Note then the head never moves left of any X, so that there are never unmatched 0's and 1's on the left of an X.
From initial state q_0, M picks up a 0 or a 1 and turns it into X. The only final state is q_4. In state q_3 M moves head left looking for the rightmost X.

	0	1	b	X	Y
q_0	(q_2, X, R)	(q_1, X, R)	(q_4, b, R)	—	(q_0, Y, R)
q_1	(q_3, Y, L)	$(q_1, 1, R)$	—	—	(q_1, Y, R)
q_2	$(q_2, 0, R)$	(q_3, Y, L)	—	—	(q_2, Y, R)
q_3	$(q_3, 0, L)$	$(q_3, 0, L)$	—	(q_0, X, R)	(q_3, Y, L)
q_4	—	—	—	—	—

E 9.7

Exercise (8.1.1 from textbook)

Give a reduction from the hello-world problem to the following problem:

given a program P and an input I, does the program ever produce any output?

solution

We modify P by making it print its output on some array A, capable of storing 12 characters. When A is full, P checks whether it stores "hello world": if it does, P prints (on the output, not on the array) some character (like @); if not, it does not print anything.

So the modified program prints some output if and only if P prints "hello, world": if we are able to determine whether a program produces any output, we can solve the hello-world problem. This ends our reduction.