Edin Guso
23435

CS407 HW3

4.6.1.:

a) $S \to ABCS \to ABCABCS \to ABCABCABCS \to ABCABCABCT_c \to ABACBCABCT_c \to ABACBACBCT_c$

$\to AABCBACBCT_c \to AABCABCBCT_c \to AABACBCBCT_c \to AAABCBCBCT_c \to AAABBCCBCT_c$

$\to AAABBCBCCT_c \to AAABBBCCCT_c \to AAABBBCCT_cc \to AAABBBCT_ccc \to AAABBBT_bccc$

$\to AAABBT_bbccc \to AAABT_bbbccc \to AAAT_abbbccc \to AAT_aabbbccc \to AT_aaabbbccc$

$\to T_aaaabbbccc \to aaabbbccc$

b) It is obvious that initially $(ABC)^n$ is generated. Later, if we do not fully perform the sorting out productions ($CA \to AC$, $BA \to AB$, $CB \to BC$), it means that some part of the string will contain either a CA, a CB or a BA. Consider each case:

- We will see BA as one of the following: BAT_a, BAT_b, BAT_c
  stuck ↑ stuck
  $\to BT_aa \to$ stuck
- We will see CA as one of the following: CAT_a, CAT_b, CAT_c
  stuck ↑ stuck
  $\to CT_aa \to$ stuck
- We will see CB as one of the following: CBT_a, CBT_b, CBT_c
  stuck, stuck
  $\to CT_bb \to$ stuck
  $\to CT_ab \to$ stuck

This means that if we do not fully sort out the string into $A^nB^nC^n$, the production will get stuck. The only productions which do not get stuck lead to $A^nB^nC^n$, which lead to $a^nb^nc^n$.

4.6.2.:

a) $G = (V, \Sigma, R, S)$ where

$V = \{S, a, b, W_L, W_R, G, L, A, B, K_1, K_2, K_3, K_4, K_5, K_6\}$,

$\Sigma = \{a, b\}$, and

$R = \{ S \to W_L G W_R,$
$G \to AG \mid BG \mid K_1 L,$
$AK_1 \to K_1 A,$
$BK_1 \to K_1 B,$
$W_L K_1 \to W_L K_2,$
$W_L K_2 A \to a W_L K_3,$
$W_L K_2 B \to b W_L K_4,$
$W_L K_2 L \to K_6,$
$K_3 A \to A K_3,$
$K_3 B \to B K_3,$
$K_3 L \to L K_3,$
$K_4 A \to A K_4,$
$K_4 B \to B K_4,$
$K_4 L \to L K_4,$
$K_3 W_R \to A K_5 W_R,$
$K_4 W_R \to B K_5 W_R,$
$A K_5 \to K_5 A,$
$B K_5 \to K_5 B,$
$L K_5 \to K_5 L,$
$W_L K_5 \to W_L K_2,$
$K_6 A \to a K_6,$
$K_6 B \to b K_6,$
$K_6 W_R \to e \}.$

Edin Guso
23435

b) $G = (V, \Sigma, R, S)$ where

$V = \{S, a, A, B, C, K_a, K_b, K_d\}$,

$\Sigma = \{a\}$,

$R = \{S \to BK_aAC \mid a$,
$K_aA \to AAK_a$,
$K_aC \to K_dC$,
$AK_d \to K_dA$,
$BK_d \to BK_a \mid K_b$,
$K_bA \to aK_b$,
$K_bC \to e\}$.

c) $G = (V, \Sigma, R, S)$ where

$V = \{S, a, W_L, W_R, G, A, B, C, D, E, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_8, K_9\}$,

$\Sigma = \{a\}$,

$R = \{S \to W_LK_1GW_R \mid e$,
$K_1G \to GABK_1$,
$K_1AB \to ABK_1$,
$K_1W_R \to K_2W_R$,
$ABK_2 \to K_2AB$,
$GK_2 \to K_2G$,
$W_LK_2 \to W_LK_1 \mid W_LK_3$,
$K_3G \to K_3$,
$K_3AB \to CDK_3$,
$K_3W_R \to K_4W_R$,
$DC \to CD$,
$DK_4 \to K_4D \mid K_5D$,
$CK_5 \to K_5C$,
$W_LK_5 \to W_LK_6$,
$K_6C \to CK_6$,
$CK_6D \to K_7D$,
$K_7D \to DEK_7$,
$K_7E \to EK_7$,
$K_7W_R \to K_8W_R$,
$CK_8 \to K_8C$,
$DK_8 \to K_8D$,
$EK_8 \to K_8E$,
$W_LK_8 \to W_LK_6$,
$W_LK_6D \to K_9D$,
$K_9D \to K_9$,
$K_9E \to aK_9$,
$K_9W_R \to e\}$.

4.6.3.:

Any production that consumes a single non-terminal can be written in the form $uAv \to uwv$ where u = LHS of the consumed nonterminal, v = RHS of the consumed nonterminal, A = consumed nonterminal, and w = generated symbols. This is trivial.

We need to convert all other productions (ones that consume ≥ 2 nonterminals). Let's solve this for the general case where the number of nonterminals consumed is n:

$uA_1 \dots A_nv \to uwv$

We can decompose this single production into following productions:

$\underbrace{u}_{u}\underbrace{A_1}_{A}\underbrace{\dots A_nv}_{v} \to \underbrace{u}_{u}\underbrace{A_2 \dots A_nv}_{v}$, $\underbrace{u}_{u}\underbrace{A_2}_{A}\underbrace{\dots A_nv}_{v} \to \underbrace{u}_{u}\underbrace{A_3 \dots A_nv}_{v}$, ..., $\underbrace{u}_{u}\underbrace{A_n}_{A}\underbrace{v}_{v} \to \underbrace{u}_{u}\underbrace{w}_{w}\underbrace{v}_{v}$

and $w = e$ and $w = e$

This way, we can convert any grammar into an equivalent grammar with rules of the from $uAv \to uwv$ where $A \in V - \Sigma$, and $u, v, w \in V^*$.

Edin Guso
23435

4.7.1.:

We can show that F is primitive recursive using induction:

Base case: $f(n)$ is primitive recursive (given)

Induction hypothesis: Let $F'(n) = f(f(\ldots f(n)\ldots))$, where there are $n-1$ function compositions be primitive recursive.

Inductive step: $F(n) = f(F'(n))$ and since we know that both f and F' are primitive recursive, then, $F(n)$ is also primitive recursive by composition.

4.7.2.:

a) $factorial(0) = one_0() = 1$; ~~for~~ $factorial(n+1) = h(n, f(n)) = (n+1) \cdot n! = (n+1)!$ where
$h(n,k) = (mult \cdot ((succ \cdot id_{2,1}), id_{2,2}))(n,k) = (n+1) \cdot k$

b) $gcd(m,0) = id_{1,1}(m) = m$; $gcd(m,n) = h(n, r, gcd(n,r))$ where
$h(n,m,k) := id_{3,3}(n,m,k) = k$ and $r :=$ remainder from dividing m by n. $\Big\} \Rightarrow$ Euclidean Algorithm

c) $prime(n) = \bigwedge_{j=2,n-1} iszero(pred(gcd(n,j)))$ for $n \geq 3$

d) $p(0) = two_0() = 2$; $p(n+1) = h(n, p(n)) = n+1^{st}$ prime number where
$h(n,k) = \mu j\, prime(j) := Min(k < j < 2^{2^{n+1}} \mid prime(j) = 1)$

e) $log(m,0) = zero_0() = 0$; $log(m,n+1) = h(m,n, log(m,n)) = \lceil log_{m+2}(n+2) \rceil$
$h(m,n,k) = plus(k, (exp(plus(m,m), k) < succ(succ(n))))$