L_1

Tuesday, October 29, 2024 1:51 PM

Devise a context-free grammar for the following language. Provide an explanation of how the grammar was devised.

- $L_1 = \{0^i 1^j : \frac{i}{i+j} \geq \frac{1}{4} \text{ and } i, j > 0\}^*$. (Please note the Kleene-* operator.)

$$\frac{i}{i+j} \geq \frac{1}{4}$$

$\Rightarrow 4i \geq i + j$ (cross multiply)

$\Rightarrow 3i \geq j$ (subtract i from both sides)

CFG grammar for the language:

S -> ST | 0S111 | T

Explanation: ST is the Kleene-* operator where T prevents an infinite loop on S; 0S111 is when 0 to 1 is in a 1:3 ratio and we split S down the middle to preserve the order of 0's and 1's

T -> 0T | 0T1 | 0T11 | 01 | 011 | 0111

Explanation: 0T, 0T1, and 0T11 is the case for 0 to up to two 1's with the possibility for more (still need to satisfy i,j>0); 01, 011, and 0111 are final cases

L_2

Thursday, October 31, 2024 11:24 AM

Devise a context-free grammar for the following language. Provide an explanation of how the grammar was devised.

- $L_2 = \{0^i 1^k 2^j 3^\ell : i - 2j = \ell - 2k \text{ and } i, j, k, \ell \geq 0\}$

$i - 2j = l - 2k$
$\Rightarrow i - l = 2j - 2k$ (swap l and $2j$)
Positive case: $i - l > 0$
Negative case: $i - l < 0$
We will examine $i - l$ and $j - k$

Positive case:

- $i - l > 0$
- $j - k > 0$

Let $\exists x: j - k = x \Rightarrow j = x + k$
Substitute $j = x + k$
$i = l + 2j - 2k = l + 2(j - k) = l + 2x$
Now, let us substitute,
$0^i 1^k 2^j 3^l$
$= 0^{2x+l} 1^k 2^{k+x} 3^l$ (replace i with $l + 2x$ and j with $k + x$)
$= 0^l 0^{2x} 1^k 2^k 2^x 3^l$ (use algebra rules of exponents to break them apart)

Negative case:

- $i - l < 0$
- $j - k < 0$

$\exists y: k - j = y \Rightarrow k = y + j$
Substitute $k = y + j$
$l = i + 2k - 2j = i + 2(k - j) = i + 2y$
Now, let us substitute,
$0^i 1^k 2^j 3^l$
$= 0^i 1^{y+j} 2^j 3^{2y+i}$ (replace l with $i + 2y$ and k with $y + j$)
$= 0^i 1^y 1^j 2^j 3^{2y} 3^i$ (use algebra rules of exponents to break them apart)

CFG grammar for the language:
S -> A | D
A -> 0A3 | B

Explanation: 0A3 accounts for 0 and 3 pairings of i, A allows us to move to handle the y pairings

B -> 1B33 | C

Explanation: 1B33 accounts for 1 and 33 pairings of y, B allows us to move to handle the j pairings

C -> 1C2 | λ

Explanation: 1C3 accounts for 1 and 3 pairings of j and λ offers finality (anecdote: we could do S -> A -> B -> C -> λ)

D -> 0D3 | E

Explanation: 0D3 accounts 0 and 3 pairings of l, A allows us to move to handle the x pairings

E -> 00E2 | F

Explanation: 00E2 accounts for 00 and 2 pairings of x, B allows us to move to handle the k pairings

F -> 1F2 | λ

Explanation: 1F2 accounts for 1 and 2 pairings of k and λ offers finality (anecdote: we could do S -> D -> E -> F -> λ)

L_3

Thursday, October 31, 2024 11:24 AM

Devise a context-free grammar for the following language. Provide an explanation of how the grammar was devised.

- $L_3 = \{0^i 1^m 2^k : i, m, k \geq 0 \text{ and } \max\{2i, 3k\} \geq 5m\}$
  Hint: $\max\{a, b\} \geq c$ if and only if $a \geq c$ or $b \geq c$.

If [two times the number of 0's (i) OR three times the number of 2's (k)] is greater than five times the number of 1's (m) AND the number of 0's, 1's, and 2's is greater than or equal to 0, then we satisfy the bounding conditions.

We have two cases for the max, we handle both of these cases with S:
S -> A | D

Case when $2i \geq 5m$:
A -> 00000A11 | B (2:5 ratio of 1's to 0's)
B -> 0B | 00B | 000B | 000B1 | 0000B | 0000B1 | 00000B | 00000B1 | C (all combinations when $2i > 5m$)
C -> C2 | λ (can recurse with 2's)

Case when $3k \geq 5m$:
D -> 111D22222 | E (3:5 ratio of 1's to 2's)
E -> E2 | E22 | 1E22 | E222 | 1E222 | E2222 | 1E2222 | 11E2222 | E22222 | 1E22222 | 11E22222 | F (all combinations when $3k > 5m$)
F -> 0F | λ (can recurse with 0's)

Combine all of these to create the CFG grammar for the language:
S -> A | D
A -> 00000A11 | B (2:5 ratio of 1's to 0's)
B -> 0B | 00B | 000B | 000B1 | 0000B | 0000B1 | 00000B | 00000B1 | C (all combinations when $2i > 5m$)
C -> C2 | λ (possible to terminate with 2's)
D -> 111D22222 | E (3:5 ratio of 1's to 2's)
E -> E2 | E22 | 1E22 | E222 | 1E222 | E2222 | 1E2222 | 11E2222 | E22222 | 1E22222 | 11E22222 | F (all combinations when $3k > 5m$)
F -> 0F | λ (possible to begin with 0's)

L_4

Thursday, October 31, 2024 11:24 AM

Devise a context-free grammar for the following language. Provide an explanation of how the grammar was devised.

- $L_4 = \{w \in \{0,1\}^* : \#_0(w) = \#_1(w) + 2\}$.
  Here, $\#_0(w), \#_1(w)$ represent the number of zeros and ones in w, respectively.

For any number of 1's, I need at least 2 more 0's. By this logic, my minimal case should be 00 (with zero number of 1's).

CFG grammar for the language:
S -> S10 | 0S1 | 01S | 10S| 1S0 | S01 | 00

Explanation: 00 is needed to stop, all other cases are combinations of recursive 0's and 1's that are balanced (that is, an equal number of 0's and 1's is admitted while still attempting further recursion)

L_5

Thursday, October 31, 2024 11:24 AM

Devise a context-free grammar for the following language. Provide an explanation of how the grammar was devised.

- $L_5 = \{w \in (\{0\} \circ \{1\}^+)^* : w$ contains two runs of ones that are of different lengths$\}$
  For the purpose of this question, please treat a run as a maximal substring of the same bit (one in this case). For example, $01^4, 01^5 01^5 01^5 \notin L$ since the former does not have two runs of ones even and the three runs of ones in the latter are each of length 5. On the other hand, $01^3 01^5, 01^7 01^9 01^9, 01011 \in L$. Note that L is infinite. Just as an aside, L_5 can be considered as the set of all lists of positive numbers that have at least two unequal list elements.

Clarifying that $\{1\}^+$ refers to at least one or more number of 1's.

We consider strings with exactly 2 runs of 1 with 1 different length.

Our minimal cases can be handled from the start:
S -> 0A | 0C

Next, let us handle two cases, one case where there's more 1's to the left of the string and one case where there's more 1's to the right of the string.

Case when the number of 1's to the left of the string is greater than the number of 1's to the right of the string:
A -> 1A1 | B
B -> 1B | 1101 (where 1101 is our minimal case)

Case when the number of 1's to the right of the string is greater than the number of 1's to the left of the string:
C -> 1C1 | D
D -> D1 | 1011 (where 1011 is our minimal case)

E is the possibility of longer strings that still satisfy the language. Therefore, we can use this consideration for our start:
S -> 0AE | 0CE | E0A | E0C
Where E is:
E -> EE | E1 | 01

Combine all of these to create the CFG grammar for the language:
S -> 0A | 0C | 0AE | 0CE | E0A | E0C
A -> 1A1 | B
B -> 1B | 1101
C -> 1C1 | D
D -> D1 | 1011
E -> EE | E1 | 01