Languages and Machines

Chap 10: 1,2,4,5

1

a

S -> aAbBab | .\
A-> aAA' | .\
B -> bBB' | .\
A'a -> aa
B'b -> bb
A'b -> bA'
B'a -> aB'

b

S -> aAbcd | .\
A->aABCD | .\
Bb->bb
Dc->cd
Dd->dd
Cb->bC
Cc-> cc

c

S -> WW'W'' | .\
W-> aA'A''W | bB'B''W | .\
W' -> .\
W'' -> .\
A'a -> aA'
A'W -> WA'
A'W' -> W'a
the same for B'
A''W -> WA''
A''W'->W'A''
A''W''->W''a
A''a->aA''
the same for B''

2

(I know this could be a lot better if I were to generalize the sentential forms and be clearer about induction. I'll come back to it if I find the time). S adds one a, one A, one b, and one c. This has the form $a^1b^1c^1$ without A. A adds one a, one A, one b, and one C. A holds the ordering for the a's and b's, since the new a's are added at the end of a^i and the new b's at the beginning of b^i. The A production always adds an equal number of a's, b's, and C's. The C's are inserted after every new added b, to make the form 'bCbCbC...'. Since we know that the a's, b's and C/c's must be equal in number, and that the a's have the correct ordering, we must show that the b's and c's end up with the correct ordering. The production Cb->bC propogates all the C's from the form 'bCbCbC...' to 'bbbb...CCCC...', which is the correct ordering. All the C's will in turn produce one new c, which ultimately produces the form $a^i b^i c^i$.

4

a

Define G = {V1 U V2 U {S}, Sigma1 U Sigma2, P1 U P2 U {S-> S1 | S2}, S). A string w is in L(G) if, and only if, there is a derivation S => S1/2 =>* w. Thus w is in L1 or L2.

b

To prove this, we can show that there is a TM that would accept strings in both languages, since a TM accepter can be built for any context-free language. Let us build a TM for L1 and another for L2. Let us build a third TM3 that runs TM1 followed by TM2. The intersection of L1 and L2 is accepted by TM3.

c

Define G = (V1 U V2 U {S}, Sigma1 U Sigma2, P1 U P2 U {S->S1S2}, S). A string w is in L3, the concatenation of L1 and L2, if it can be derived from the form S=>S1S2=>*uS2=>*uv, where u is an element of L1 and v is an element of L2.

d

Define G = (V1, Sigma1, P1 U {S ->S1S | .\}, S). The S rule of G generates any number of copies of S1. Each of these, in turn, initiates the derivation of a string in L1. The concatenation of any number of strings from L1 yields L1*.

e

We can design a TM accepter that takes a string h, which is the homomorphic image of a string w in our language. Since h(w) is a Turing computable function, then we can have our TM accepter repeatedly run h(w) on all strings in L until there is a match; otherwise the machine will not halt.

5

b(ab)*
S -> bA
A -> abA | .\
d(q0, B) = [q1, B, R]
d(q1, b) = [q2, b, R]
d(q2, a) = [q1, a, R]
d(q1, b) = [q2, b, R] accepts
S => bA
=> babA
=> bab