Formal Languages Homework 10

Liam Dillingham

April 28, 2019

1 Problem 9.1.1

What strings are:

1.1 w_{37}?

37 in binary is 100101. then $1w = 100101$, so $w = 00101$.

1.2 w_{100}?

100 in binary is 1100100. then $1w = 1100100$, so $w = 100100$.

2 Problem 9.2.1

Show that the halting problem, the set of (M, w) pairs such that M halts (with or without accepting) when given input w is r.e. but not recursive

---

For a given input w, the TM M must decide whether or not the string is in the language. Let L_u be the set of pairs of (M, w) such that M halts on w. By encoding M, then we can rename our set X to L_u, that is, $L_u(M, w) = \{(M, w) \mid M \text{ halts on } w\}$. Then $\overline{L_u}$ is the set of pairs of (M, w) such that M does *not* halt on w. If L_u is recursive, then all pairs of (M, w) are in L_u, since a recursive language always halts. However, $\overline{L_u}$ would have to be recursive as well, since the compliment of a recursive language must also be recursive. Yet $\overline{L_u}$ is precisely the language of the pairs of (M, w) which do not halt. Since a recursive language must halt eventually, then determining the set of pairs which halt is recursievely enumerable, since determining which ones do not halt requires infinite computation.

3 Problem 9.2.2

Using *Ackermann's function*:

1. $A(0, y) = 1$ for any $y \geq 0$
2. $A(1, 0) = 2$
3. $A(x, 0) = x + 2$ for $x \geq 2$
4. $A(x + 1, y + 1) = A(A(x, y + 1), y)$ for any $x \geq 0$, and $y \geq 0$

3.1 a). Evaluate $A(2, 1)$

$A(2, 1) = A(1 + 1, 0 + 1)$ Rule #4 $\Rightarrow A(1 + 1, 0 + 1) = A(A(1, 0 + 1), 0)$
$A(1 + 1, 0 + 1) = A(A(1, 0 + 1), 0) = A(A(1, 1), 0)$
$A(1, 1) = A(0 + 1, 0 + 1) = A(A(0, 0 + 1), 0) = A(A(0, 1), 0)$
$A(0, 1) = 1$. Now substitute
$A(1, 1) = A(0 + 1, 0 + 1) = A(A(0, 1), 0) = A(1, 0) = 2$. And substituting again,
$A(2, 1) = A(1 + 1, 0 + 1) = A(A(1, 0 + 1), 0) = A(A(1, 1), 0) = A(2, 0) = x + 2 = 4$

3.2 b). What function of x is $A(x, 2)$?

3.3 c). Evaluate $A(4, 3)$

$A(4, 3) = A(3 + 1, 2 + 1) = A(A(3, 2 + 1), 2)$
$A(3, 2 + 1) = A(3, 3) = A(2 + 1, 2 + 1) = A(A(2, 2 + 1), 2)$**
$A(2, 2 + 1) = A(2, 3) = A(1 + 1, 2 + 1) = A(A(1, 2 + 1), 2)$
$A(1, 2 + 1) = A(1, 3) = A(0 + 1, 2 + 1) = A(A(0, 2 + 1), 2)$*
$A(0, 2 + 1) = A(0, 3) = 1$. Now we subtitute backwards.
$A(A(0, 2 + 1), 2) = A(A(0, 3), 2) = A(1, 2)$.
$A(1, 2) = A(0 + 1, 1 + 1) = A(A(0, 1 + 1), 1)$
$A(0, 1 + 1) = A(0, 2) = 1$. Subtitute back.
$A(1, 2) = A(A(0, 2), 1) = A(1, 1)$. Using results derived from previous questions, $A(1, 1) = 2 = A(1, 2)$. Substitute back starting at line *.
$A(1, 3) = A(A(0, 3), 2) = A(1, 2) = 2$
$A(2, 3) = A(A(1, 3), 2) = A(2, 2)$
$A(2, 2) = A(1 + 1, 1 + 1) = A(A(1, 1 + 1), 1)$
$A(1, 1 + 1) = A(1, 2) = 2$. subtitute
$A(2, 2) = A(2, 1) = 4$. (From previous question). Now substitute back to **. $A(3, 3) = A(A(2, 3), 2)$. Since $A(2, 2) = A(2, 3) = 4$, then $A(3, 3) = A(4, 2)$.
$A(4, 2) = A(3 + 1, 1 + 1) = A(A(3, 1 + 1), 1)$
$A(3, 1 + 1) = A(3, 2) = A(2 + 1, 1 + 1) = A(A(2, 1 + 1), 1)$
$A(2, 1 + 1) = A(2, 2) = 4$. substitute
$A(3, 2) = A(4, 1)$.

I want to say that at this point I got tired of doing the calculations by hand, so I wrote a python script which computes the Ackermann function. However, there seemed to be some sort of recursion error with the code at $A(4, 3)$ so I computed it for $A(4, 1)$ and got 8.

$A(4, 1) = 8$. Substitute back
$A(4, 2) = A(A(3, 2), 1) = A(8, 1)$. Using our script, we get $A(8, 1) = 16$. So $A(4, 2) = A(8, 1) =$

$16 = A(3, 3)$.
$A(4, 3) = A(16, 2)$

The python code is shown below:

```
import sys
sys.setrecursionlimit(8000)

def A(x, y):

    # rule 1
    if x == 0 and y >= 0:
        return 1

    # rule 2
    if x == 1 and y == 0:
        return 2

    # rule 3
    if x >= 2 and y == 0:
        return x + 2

    # rule 4
    if x >= 0 and y >= 0:
        x = x - 1
        y = y - 1

        return A(A(x, y + 1), y)

    raise Exception('Something terrible happened!')

print(A(16,2))
```