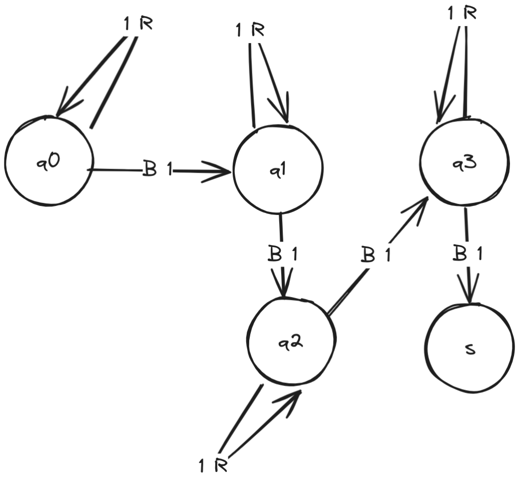
HW 08

Elizabeth Hunt (A02364151)

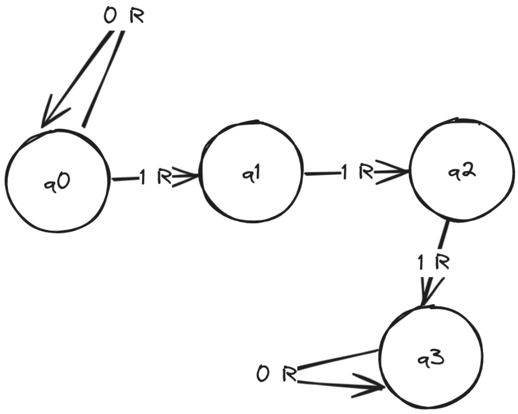
November 17, 2023

Contents

1 Problem One



2 Problem Two



3 Problem Three

Using the following code proceeding the appendix we receive
$l(111) = 4$
$r(111) = 3$
$lt(111) = 12$

```
const p3 = () => {
  const x = 111;
  const { l, r } = lr(x);
  const lt = length(x);

  [‘l(${x}) = ${l}‘, ‘r(${x}) = ${r}‘, ‘lt(${x}) = ${lt}‘].forEach((s) =>
    console.log(s)
  );
};
p3();
```

4 Problem Four

Using the following code proceeding the appendix we receive
$(17)_0 = 0$
$(17)_1 = 0$
$(17)_2 = 0$
$(17)_3 = 0$
$(17)_4 = 0$
$(17)_5 = 0$
$(17)_6 = 0$
$(17)_7 = 1$
$(17)_8 = 0$

```
const p4 = () => {
  const x = 17;

  for (let i = 0; i <= 8; i++)
    console.log(‘(${x})_${i} = ${access(x, i)}‘)
```

```
};
p4();
```

And for all $i > 8$, $p_i > 17$ and thus $p_i^{(t+1)} \nmid x$ for all $t \geq 0$, and thus the valid set of t's, T, has $\min(T) = 0$, so $(17)_i = 0$.

5 Problem Five

We compute the new code:
$[\#(I_1), \#(I_2)]$
For $\#(I_1)$:
$\#(I_1) = \langle a, \langle b, c\rangle\rangle$ where $a = \#(B1) = 2$, $b = 1$, $c = \#(X1) - 1 = 1$, so $\#(I_1) = \langle 2, \langle 1, 1\rangle\rangle = 2^2(2\langle 1, 1\rangle + 1) - 1 = 4(11) - 1 = 43$.
For $\#(I_2)$:
$\#(I_2) = \langle a, \langle b, c\rangle\rangle$ where $a = 0$ as there is no label for the instruction, $b = \#(B1) + 2 = 4$, $c = \#(X1) - 1 = 1$, so $\#(I_2) = \langle 0, \langle 4, 1\rangle\rangle = 2^0(2\langle 4, 1\rangle + 1) - 1 = 94$.
Thus:
$[43, 94] = (2^{43})(3^{94}) - 1 = 621853033458669921161454887237476225967217597213819745075 1$.

6 Problem Six

6.1 $\phi_5{}^1$(x)

$\phi_5{}^1$(x) has source $5 + 1 = 6$ which corresponds to the godel sequence $2^1 * 3^1 = [1, 1]$. $1 = \langle 1, \langle 0, 0\rangle\rangle$ which corresponds to an instruction with $\#(L) = 1$, $\#(V) = 0$, and an operation of 0:

```
[ A1 ] Y <- Y
[ A1 ] Y <- Y
```

6.2 $\phi_7{}^1$(x)

$\phi_7{}^1$(x) has source $7 + 1 = 8$ which corresponds to the godel sequence $2^3 = [3]$. $3 = \langle 2, \langle 0, 0\rangle\rangle$ which corresponds to an instruction with $\#(L) = 2$, $\#(V) = 0$, and an operation of 0:

```
[ B1 ] Y <- Y
```

6.3 $\phi_{11}{}^1$(x)

$\phi_{11}{}^1$(x) has source $11 + 1 = 12$ which corresponds to the godel sequence $2^2 * 3^1 = [2, 1]$. $2 = \langle 0, \langle 1, 0\rangle\rangle$ which corresponds to an instruction with $\#(L) = 0$, $\#(V) = 0$, and an operation of 1. And, we already found 1 in $\phi_5{}^1$(x):

```
Y <- Y + 1
[ A1 ] Y <- Y
```

6.4 $\phi_{13}{}^1$(x)

$\phi_{13}{}^1$(x) has source $13 + 1 = 14$ which corresponds to the godel sequence $2^1 * 7^1 = [1, 0, 0, 1]$. And, we already found 1 in $\phi_5^1(x)$, 0 is trivially `Y <- Y` (unlabeled, $\#(V) = 0$, op $= 0$).

```
[ A1 ] Y <- Y
Y <- Y
Y <- Y
[ A1 ] Y <- Y
```

6.5 $\phi_{17}{}^1$(x)

ϕ_{17}(x) has source $17 + 1 = 18$ which corresponds to the godel sequence $2^1 * 3^2 = [1, 2]$. And, we already found 1 in $\phi_5^1(x)$, and 2 in $\phi_{11}{}^1$(x)

```
[ A1 ] Y <- Y
Y <- Y + 1
```

7 Problem Seven

1. Let $m(x_1, x_2) = x_1 * x_2$ which is primitive recursive by proofs in class.
2. Let $four(x_1) = s(s(s(s(n(x_1)))))$; the successor function composed 4 times on the null function.
3. Then $f(x_1)$ is $m(four(x_1), x_1)$ which is an application of composition of primitive recursive functions.

Thus, f is primitive recursive, and thus computable.

8 Problem Eight

We can use the handy identity that $lcm(x_1, x_2) = \frac{(a*b)}{gcd(a,b)} = \lfloor \frac{(a*b)}{gcd(a,b)} \rfloor$ since $gcd(a, b)$ by definition divides $a * b$.

1. Define $gcd(x_1, 0) = x_1$
2. Let $R(x_1, x_2)$ be the remainder function when x_1 divides x_2 which is primitive recursive by the proof found on page 56 of the book.
3. We construct the informal recursion $gcd(x_1, x_2) = gcd(x_2, R(x_1, x_2))$ by Euclid's Algorithm.
4. Let $floordiv(x_1, x_2)$ be the floor of the result of division $\frac{x_1}{x_2}$ which is primitive recursive by the proof found on page 56 of the book.
5. Let $m(x_1, x_2) = x_1 * x_2$ which is primitive recursive by proofs in class.
6. Then $lcm(x_1, x_2) = floordiv(m(x_1, x_2), gcd(x_1, x_2))$ which is an application of composotion of primitive recursive functions.

Then lcm is primitive recursive.

9 Appendix

```
const isPrime = (n) =>
  !Array(Math.ceil(Math.sqrt(n)))
    .fill(0)
    .map((_, i) => i + 2) // first prime is 2
    .some((i) => n !== i && n % i === 0);

const primesCache = [2];
const p = (i) => {
  if (primesCache.length <= i) {
```

```
    let x = primesCache.at(-1);
    while (primesCache.length <= i) {
      if (isPrime(++x)) primesCache.push(x);
    }
  }
  return primesCache.at(i - 1);
};

const lr = (z, maxSearch = 100) => {
  let x = 0;
  for (let i = 0; i < maxSearch; ++i)
    if ((z + 1) % Math.pow(2, i) === 0) x = Math.max(i, x);

  const y = ((z + 1) / Math.pow(2, x) - 1) / 2;
  return { l: x, r: y };
};

const access = (x, i) => {
  if (i === 0 || x === 0) return 0;

  const p_i = p(i);
  let minT = x;
  for (let t = x; t >= 0; t--)
    if (x % Math.pow(p_i, t + 1) !== 0) minT = Math.min(t, minT);
  return minT;
};

const length = (x) => {
  let minI = x;
  for (let i = x; i >= 0; i--)
    if (
      access(x, i) !== 0 &&
      Array(x)
        .fill(0)
        .map((_, j) => j + 1)
        .every((j) => j <= i || access(x, j) == 0)
    )
      minI = Math.min(i, minI);

  return minI;
};
```