Assignment 2 - Emirps

Rahil Agrawal z5165505
Aditya Karia zXXXXXXX

COMP2111 18s1

1 Task 1 - Specification Statement

Notes,x:
-Write neatly
-make sure grammar is correct
-look at examples for default spec structure.

Define an Emirp using 2 functions - reverse and prime. Make these functions match with their given specs in order to help prove implications.

Pre condition,x: n is a positive number - $n > 0$

Post condition $EMIRP(r, n)$ where r is the n^{th} emirp(where emirp is as defined above). Therefore our program can be specified by,x:
proc EMIRP(**value** n, **result** r) $\cdot$
$\llcorner n, r, x : \left[\ n > 0, Emirp(r, n)\ \right]\lrcorner_{(1)}$

2 Task 2 - Derivation

$$
\begin{array}{ll}
& \textbf{proc } \text{EMIRP}(\textbf{value } n, \textbf{result } r) \cdot \llcorner n, r, x : [\ n > 0, Emirp(r,n)\ ]\lrcorner_{(1)} \\
(1) \sqsubseteq & \quad \langle \textbf{c-frame} \rangle \\
& \llcorner r, x : [\ n > 0, Emirp(r,n)\ ]\lrcorner_{(2)} \\
(2) \sqsubseteq & \quad \langle \textbf{i-loc} \rangle \\
& \llcorner i, x, r, x : [\ n > 0, Emirp(r,n)\ ]\lrcorner_{(3)} \\
(3) \sqsubseteq & \quad \langle \textbf{seq} \rangle \\
& \llcorner i, x, r, x : [\ n > 0, i = 1 \wedge x = 13 \wedge n > 0\ ]\lrcorner_{(4)}; \\
& \llcorner i, x : [\ i = 1 \wedge x = 13 \wedge n > 0, Emirp(r,n)\ ]\lrcorner_{(5)} \\
(4) \sqsubseteq & \quad \langle \textbf{c-frame} \rangle \\
& i, x, x : [\ n > 0, i = 1 \wedge x = 13 \wedge n > 0\ ] \\
\sqsubseteq & \quad \langle \textbf{ass - (1)} \rangle \\
& i := 1 \\
& x := 13 \\
(5) \sqsubseteq & \quad \langle \textbf{seq} \rangle \\
& \llcorner i, r, x : [\ i = 1 \wedge n > 0, Inv\ ]\lrcorner_{(6)}; \\
& \llcorner i, r, x : [\ Inv, Inv \wedge i = n\ ]\lrcorner_{(7)}; \\
& \llcorner i, r, x, x : [\ Inv \wedge i = n, Emirp(r,n)\ ]\lrcorner_{(8)} \\
(6) \sqsubseteq & \quad \langle \textbf{w-pre, c-frame - (2)} \rangle \\
& \llcorner r, x : [\ Inv[^{13}/_{x}], Inv\ ]\lrcorner_{(9)} \\
\sqsubseteq & \quad \langle \textbf{ass - (3)} \rangle \\
& r := 13 \\
(7) \sqsubseteq & \quad \langle \textbf{while} \rangle \\
& \textbf{while } i \neq n \textbf{ do} \\
& \quad \llcorner i, r, x : [\ Inv \wedge i \neq n, Inv\ ]\lrcorner_{(10)} \\
& \textbf{od}; \\
(8) \sqsubseteq & \quad \langle \textbf{w-pre, c-frame} \rangle \\
& \llcorner r, x : [\ \text{EMIRP}(r,n), \text{EMIRP}(r,n)\ ]\lrcorner_{(11)} \\
\sqsubseteq & \quad \langle \textbf{skip - (4)} \rangle \\
& skip \\
(10) \sqsubseteq & \quad \langle \textbf{seq} \rangle \\
& \llcorner r, x : [\ Inv \wedge i \neq n, Inv[^{x+1}/_{x}]\ ]\lrcorner_{(11)}; \\
& \llcorner r, x : [\ Inv[^{x+1}/_{x}], Inv\ ]\lrcorner_{(12)}
\end{array}
$$

$(11) \sqsubseteq \quad \langle \textbf{ass - (5)} \rangle$

$x := x + 1$

$(12) \sqsubseteq \quad \langle \textbf{i-loc, seq} \rangle$

$\llcorner a, i, r, x : [\ Inv[^{x+1}/_x], Inv[^{x+1}/_x] \wedge a = 1\ ]\lrcorner_{(14)};$

$\llcorner a, i, r, x : [\ Inv[^{x+1}/_x] \wedge a = 1, Inv\ ]\lrcorner_{(15)}$

$(14) \sqsubseteq \quad \langle \textbf{c-frame} \rangle$

$a, x : [\ Inv[^{x+1}/_x], Inv[^{x+1}/_x] \wedge a = 1\ ]$

$\sqsubseteq \quad \langle \textbf{ass - (6)} \rangle$

$a := 1$

$(15) \sqsubseteq \quad \langle \textbf{seq} \rangle$

$$\llcorner a, i, r, x : \left[\begin{array}{l} Inv[^{x+1}/_x] \wedge a = 1, (a = 1\ \wedge \neg\exists k \in 2..(x-1)\,(x \textbf{ mod } k = 0)) \\ \vee(a = 0 \wedge \exists k \in 2..(x-1)\,(x \textbf{ mod } k = 0))) \end{array} \right]\lrcorner_{(16)};$$

$$\llcorner a, i, r, x : \left[\begin{array}{l} (a = 1\ \wedge \neg\exists k \in 2..(x-1)\,(x \textbf{ mod } k = 0)) \\ \vee(a = 0 \wedge \exists k \in 2..(x-1)\,(x \textbf{ mod } k = 0))), Inv \end{array} \right]\lrcorner_{(17)}$$

$(16) \sqsubseteq \quad \langle \textbf{w-pre} \rangle$

$a, x : [\ a > 1 \wedge x > 0, post(16)\ ]$

$\sqsubseteq \quad \langle \textbf{proc} \rangle$

$isPrime(x, a)$

$(17) \sqsubseteq \quad \langle \textbf{if} \rangle$

if $a = 1$

then $\llcorner a, i, r, x : [a = 1 \wedge pre(18), post(18)]\lrcorner_{(18)}$

else $\llcorner p, x : [a \neq 1 \wedge pre(18), post(18)]\lrcorner_{(19)}$

fi

$(18) \sqsubseteq \quad \langle \textbf{i-loc} \rangle$

$a, i, r, s, x : [\ pre(18), post(18)\ ]$

$\sqsubseteq \quad \langle \textbf{seq} \rangle$

$\llcorner s, x : [\ pre(18), post(18) \wedge s = 0\ ]\lrcorner_{(20)};$

$\llcorner a, i, r, s, x : [\ post(18) \wedge s = 0, post(18)\ ]\lrcorner_{(21)}$

$(20) \sqsubseteq \quad \langle \textbf{ass - (7)} \rangle$

$s := 0$

$(21) \sqsubseteq \quad \langle \textbf{seq} \rangle$

$\llcorner s, x : [\ pre(21),$ reversen function post condition $]\lrcorner_{(22)};$

$\llcorner a, i, r, s, x : [$ reversen function post condition, $post(21)\ ]\lrcorner_{(23)}$

$(22) \sqsubseteq \quad \langle \textbf{w-pre - (8)} \rangle$

$s, x : [$ reverse function pre condition, reversen function post condition $]$

$\sqsubseteq \quad \langle \textbf{proc} \rangle$

$reversen(x, s)$

$(23) \sqsubseteq \quad \langle \textbf{i-loc, seq} \rangle$

$\llcorner a, i, r, s, b, x : [\ pre(23), pre(23) \wedge b = 1\ ]\lrcorner_{(24)};$

$\llcorner a, i, r, s, b, x : [\ pre(23) \wedge b = 1, post(23)\ ]\lrcorner_{(25)}$

$(24) \sqsubseteq \quad \langle \textbf{c-frame, ass - (9)} \rangle$

$b := 1$

$(25) \sqsubseteq \quad \langle \textbf{seq} \rangle$

$\llcorner a, i, r, s, b, x : [\ pre(23) \wedge b = 1, ...prime..\ ]\lrcorner_{(26)};$

$\llcorner a, i, r, s, b, x : [\ ..prime.., post(23)\ ]\lrcorner_{(27)}$

We gather the code for the procedure body of EMIRP,x:

We have derived our code. However we need to prove **some** refinements.

2.1 Implication 1: $(4) \sqsubseteq i := 1$

$K \wedge i \geq 1 \wedge a[i][l] = a[i-1][l] \wedge j = 1 \wedge a[i][l] = 0$

$\Rightarrow \langle K \wedge B \wedge C \wedge D \wedge E \Rightarrow K \wedge C \wedge E.\text{Expanding K} \rangle$

$I \wedge j \in (0, 1) \wedge l \in 0..y, a[i][y] = 0 \wedge \forall w \in 0..(l-1)\,(a[i][w] = a[i-1][w]) \wedge$
$a[i][l] = a[i-1][l] \wedge a[i][l] = 0$

$\Rightarrow \langle \text{since true for } w \text{ in 0..l-1 and also for l} \rangle$

$I \wedge j \in (0, 1) \wedge l \in 0..y, a[i][y] = 0 \wedge \forall w \in 0..l\,(a[i][w] = a[i-1][w] \wedge a[i][l] = 0)$

$\Rightarrow \langle \text{separating for } l \text{ in 0..y-1 and l=y} \rangle$

$I \wedge j \in (0, 1) \wedge l \in 0..y-1, a[i][y] = 0 \wedge \forall w \in 0..l\,(a[i][w] = a[i-1][w]) \wedge a[i][l] = 0 \vee$
$I \wedge j \in (0, 1) \wedge l = y \wedge a[i][l] = 0, a[i][y] = 0 \wedge \forall w \in 0..l\,(a[i][w] = a[i-1][w])$

$\Rightarrow \langle \text{since a[i][l] = a[i][y] = 0 when l=y} \rangle$

$I \wedge j \in (0, 1) \wedge l \in 0..y-1, a[i][y] = 0 \wedge \forall w \in 0..l\,(a[i][w] = a[i-1][w]) \vee TRUE$

$\Leftrightarrow \langle \text{definitions of } I \text{ and substitution , } l \text{ in 0..y-1 means (l+1) in 0..y} \rangle$

$K[^{l+1}/_l][^0/_j]$

3 Task 3 - C Code

```
#include <stdio.h>
#include "reverse.h"
```

```

unsigned long emirp(unsigned long n);
void isPrime(unsigned long r, int *a);

int main (int argc, char* argv[]){
        unsigned long n;
        if(scanf("%lu", &n)==1)
            printf("%lu\n",emirp(n));
}

/*
var i := 1
r := 13
while i != n do
    r := r + 1
    var a := 1
    isPrime(r,a)
    if a = 1 then
        var s := 0
        reversen(r,s)
        var b := 1
        isPrime(s, b)
        if b = 1 && s != r then
            i = i + 1
od

*/
unsigned long emirp(unsigned long n) {
        int i = 1;
        unsigned long r = 13;
    while (i != n) {
        r = r + 1;
        int a = 1;
        isPrime(r, &a);
        if (a == 1) {
            unsigned long s = 0;
            reversen(r, &s);
            int b = 1;
            isPrime(s, &b);
            if (b == 1 && s != r) {
                i = i + 1;
            }
        }
    }
```

```
        return r;
}

/*
var j := 0
while j != r do
    if r mod j = 0 then
        a = 0
    j := j + 1
od
*/
void isPrime(unsigned long r, int *a) {
    unsigned long j = 2;
    while (j != r) {
        if (r % j == 0) {
            *a = 0;
        }
        j = j + 1;
    }
}
```

- Write something about how the C code relates.
- Compare with examples