Assignment 2

Ruofei HUANG(z5141448)   Anqi ZHU(z5141541)

May 1, 2018

1 Task 1

1.1 Prime

We define a number n to be a prime if it is a natural number greater than 1 and cannot be formed by mutiplying two natural numbers (bigger than 1) smaller than itself[1]. Hence we can describe the set containing all primes as:

$$Prime = \{n \in \mathbb{N} | \neg(\exists x \in (1..n-1)\,(x|n)) \wedge n > 1\}$$

GMP provides a function called ISPRIME() to check if a certain number n is a prime or not. The procedure of this function can be expressed as:

$$\textbf{proc } ISPRIME(\textbf{value } n, \textbf{result } p)\cdot$$
$$n, p : [TRUE, (n_0 \in Prime \wedge p > 0) \vee (n_0 \notin Prime \wedge p \leq 0)]$$

1.1.1 Procedure Call Simplification

We shall simplify its procedure call sugar, **result** $p = ISPRIME(\textbf{value } n)$, to verify primes later in our refinement.
According to its original procedure call sugar, we can explain the verification in toy

[1] Reference from Wikipedia: https://en.wikipedia.org/wiki/Prime_number

language as:

var c;
$ISPRIME(n, c)$;
if $c > 0$ **then**
  $\{n \in Prime\}$
  ...
else
  $\{n \notin Prime\}$
  ...
fi

This can be replaced by our homemade procedure call sugar ~:

if $ISPRIME(n) > 0$ **then**
  $\{n \in Prime\}$
  ...
else
  $\{n \notin Prime\}$
  ...
fi

to simplify our proof. We can also use this simplified sugar in while rules as:

while $ISPRIME(n) > 0$ **do**
  $\{n \in Prime\}$
  ...
od

1.2 Reverse

By the spec in verifying a number v which is the reverse of the number n [2], we can have the following mathematical definition:

$$v = rev(n) = \sum_{i=0}^{c(n)} (S_i 10^i)$$

[2]A proof provided by Lecturer in Control of this course on https://www.cse.unsw.edu.au/~cs2111/18s1/lec/reverse.pdf

where:

$$c(n) = \lfloor log_{10}(n) \rfloor,$$
$$S = [10]^*,$$
$$n \in \mathbb{N} \land n = \sum_{i=0}^{c(n)} (S_i 10^{(c(n)-i)})$$

Then we can simplify the spec of **proc** $reversen$(**value** $n : \mathbb{N}$, **result** $v : \mathbb{N}$) given by the lecturer, as

$$\textbf{proc } reversen(\textbf{value } n : \mathbb{N}, \textbf{result } v : \mathbb{N}) \cdot$$
$$r, v : [\text{TRUE}, v = rev(r_0)]$$

1.3 Emirp

An emirp n is a prime number that results in a different prime when its decimal digits are reversed[3]. The definition of emirp can be construct in mathematical semantics as follows:

$$n \in Emirp \iff n \in Prime \land rev(n) \in Prime \land n \neq rev(n)$$

Then we can define a function to check if the specified number n is an emirp. The function is such as:

$$isEmirp(n) = \begin{cases} 1 & \text{if } n \in Prime \land rev(n) \in Prime \land n \neq rev(n) \\ 0 & \text{else} \end{cases}$$

We use 0 and 1 as our returning value of the function, so that we can find out how many emirps are found in the range of 2 .. n according to the following mathematics semantics:

$$\textbf{the number of emirps found} = \sum_{i=0}^{n} isEmirp(i)$$

where:

$$n \in \mathbb{N}_{>1}$$

1.3.1 Derivation of ISEMIRP() Procedure Call

We want to transfer the isEmirp() function into a procedure so that we can use it in our later refinement. We start with a spec of the procedure.

$$\textbf{proc } ISEMIRP(\textbf{value } n : \mathbb{N}, \textbf{result } w) \cdot$$
$$\llcorner n, w : \left[\text{TRUE}, \begin{pmatrix} (w = 1 \land rev(n_0) \neq n_0 \land n_0 \in Prime \land rev(n_0) \in Prime) \lor \\ (w = 0 \land \neg(rev(n_0) \neq n_0 \land n_0 \in Prime \land rev(n_0) \in Prime)) \end{pmatrix}\right] \lrcorner_{(A)}$$

[3] Another reference from Wikipedia :https://en.wikipedia.org/wiki/Emirp

$(A) \sqsubseteq$ $\langle$**c-frame**$\rangle$

$$w : [\text{TRUE}, \left(\begin{array}{l} (w = 1 \wedge rev(n) \neq n \wedge n \in Prime \wedge rev(n) \in Prime) \vee \\ (w = 0 \wedge \neg(rev(n) \neq n \wedge n \in Prime \wedge rev(n) \in Prime)) \end{array} \right)]$$

$\sqsubseteq$ $\langle$**i-loc**$\rangle$

$$\mathbf{var}\ r \cdot r, w : [\text{TRUE}, \left(\begin{array}{l} (w = 1 \wedge rev(n) \neq n \wedge n \in Prime \wedge rev(n) \in Prime) \vee \\ (w = 0 \wedge \neg(rev(n) \neq n \wedge n \in Prime \wedge rev(n) \in Prime)) \end{array} \right)]$$

$\sqsubseteq$ $\langle$**seq2**$\rangle$

$\llcorner r : [\text{TRUE}, r = rev(n)];\lrcorner_{(B)}$

$$\llcorner r, w : [r = rev(n), \left(\begin{array}{l} (w = 1 \wedge rev(n) \neq n \wedge n \in Prime \wedge \\ rev(n) \in Prime) \vee (w = 0 \wedge \neg(rev(n) \neq n \\ \wedge n \in Prime \wedge rev(n) \in Prime)) \end{array} \right)]\lrcorner_{(C)}$$

$(B) \sqsubseteq$ $\langle$**proc**$\rangle$

$reversen(n, r);$

$(C) \sqsubseteq$ $\langle$**c-frame**$\rangle$

$$\llcorner w : [r = rev(n), \left(\begin{array}{l} (w = 1 \wedge rev(n) \neq n \wedge n \in Prime \wedge \\ rev(n) \in Prime) \vee (w = 0 \wedge \neg(rev(n) \neq n \\ \wedge n \in Prime \wedge rev(n) \in Prime)) \end{array} \right)]\lrcorner_{(C*)}$$

$(C*) \sqsubseteq$ $\langle$**s-post**, justified below in 1.3.2$\rangle$

$$\llcorner w : [r = rev(n), \left(\begin{array}{l} (w = 1 \wedge r \neq n \wedge n \in Prime \wedge r \in Prime) \vee \\ (w = 0 \wedge \neg(r \neq n \wedge n \in Prime \wedge r \in Prime)) \end{array} \right)]\lrcorner_{(C')}$$

$(C') \sqsubseteq$ $\langle$**if**$\rangle$

if $r \neq n$

then $\llcorner w : [r \neq n \wedge pre(C'), post(C')]\lrcorner_{(D)}$

else $\llcorner w : [r = n \wedge pre(C'), post(C')]\lrcorner_{(E)}$

fi

$(D) \sqsubseteq$ $\langle$**if**$\rangle$

if $ISPRIME(n) > 0$

then $\llcorner w : [pre(D) \wedge n \in Prime, post(C')]\lrcorner_{(F)}$

else $\llcorner w : [pre(D) \wedge n \notin Prime, post(C')]\lrcorner_{(G)}$

fi

$(F) \sqsubseteq$ $\langle$**if**$\rangle$

if $ISPRIME(r) > 0$

then$\llcorner w : [pre(F) \wedge r \in Prime, post(C')]\lrcorner_{(H)}$

else$\llcorner w : [pre(F) \wedge r \notin Prime, post(C')]\lrcorner_{(I)}$

fi

$(E)(G)(I) \sqsubseteq$ $\langle$**ass**, justfied below in 1.3.3$\rangle$

$w := 0$

$(H) \sqsubseteq$ $\langle$**ass**, justified below in 1.3.4$\rangle$

$w := 1$

We gather the code for the procedure body of ISEMIRP:

$$
\begin{array}{l}
\textbf{var } r; \\
reversen(n, r); \\
\textbf{if } r \neq n \textbf{ then} \\
\quad \textbf{if } ISPRIME(n) > 0 \textbf{ then} \\
\quad\quad \textbf{if } ISPRIME(r) > 0 \\
\quad\quad \textbf{then } w := 1; \\
\quad\quad \textbf{else } w := 0; \\
\quad\quad \textbf{fi} \\
\quad \textbf{else } w := 0; \\
\quad \textbf{fi} \\
\textbf{else } w := 0; \\
\textbf{fi}
\end{array}
$$

1.3.2 Proof of $r = rev(n)[^{w_0}/_w] \wedge post(C*)$

$$
\begin{array}{ll}
& r = rev(n)[^{w_0}/_w] \wedge post(C') \\
\Leftrightarrow & \quad \langle \text{Subutitue and expand } post(C') \rangle \\
& r = rev(n) \wedge \left(\begin{array}{l} (w = 1 \wedge rev(n) \neq n \wedge n \in Prime \wedge \\ rev(n) \in Prime) \vee (w = 0 \wedge \neg(rev(n) \neq n \\ \wedge n \in Prime \wedge rev(n) \in Prime)) \end{array} \right) \\
\Leftrightarrow & \quad \langle \text{Substitute } r = rev(n) \rangle \\
& \left(\begin{array}{l} (w = 1 \wedge r \neq n \wedge n \in Prime \wedge r \in Prime) \vee \\ (w = 0 \wedge \neg(r \neq n \wedge n \in Prime \wedge r \in Prime)) \end{array} \right) \\
\Leftrightarrow & \quad \langle \text{definition of } post(C) \rangle \\
& post(C)
\end{array}
$$

1.3.3 Proof of $(E)(G)(I) \sqsubseteq w := 0$

Before proving this assertion, we oberserve that in the $post(C')$, we have

$$
\begin{array}{l}
\quad \neg(r \neq n \wedge n \in Prime \wedge r \in Prime) \\
\Leftrightarrow r = n \vee n \notin Prime \vee r \notin Prime \\
\Rightarrow w = 0
\end{array}
$$

Then this assertion is valid:

$$r = n \vee n \notin Prime \vee r \notin Prime \Rightarrow post(C')[^0/_w]$$

By observe the $pre(E)$,$pre(G)$,$pre(I)$, we have:

$$pre(E) \Rightarrow r = n \Rightarrow post(C')[^0/_w]$$
$$pre(G) \Rightarrow n \notin Prime \Rightarrow post(C')[^0/_w]$$
$$pre(I) \Rightarrow r \notin Prime \Rightarrow post(C')[^0/_w]$$

Which all prove the validity.

1.3.4 Proof of $(H) \sqsubseteq w := 1$

Similar to 1.3.3 we have:

$$pre(H) \Leftrightarrow (r \neq n \wedge n \in Prime \wedge r \in Prime) \Rightarrow post(C')[^1/_w]$$

1.4 Specification of the Main Procedure

The job of our main procedure, **proc** $EMIRP($**value** $n : \mathbb{N},$ **result** $r)$, is to find and return the n^{th} emirp, where n is a given positive parameter. Using the function $isEmirp()$ which is defined and proved above, we can specify our main procedure as:

$$\textbf{proc } EMIRP(\textbf{value } n : \mathbb{N}, \textbf{result } r) \cdot$$
$$\llcorner n, r : [n > 0, \sum_{i=2}^{r} isEmirp(i) = n_0 \wedge r \in Emirp]\lrcorner_{(1)}$$

2 Task 2

$$(1) \sqsubseteq \quad \langle\textbf{c-frame}\rangle$$
$$r : [n > 0, \sum_{i=2}^{r} isEmirp(i) = n \wedge r \in Emirp]$$
$$\sqsubseteq \quad \langle\textbf{i-loc}\rangle$$
$$\textbf{var } count \cdot count, r : [n > 0, \sum_{i=2}^{r} isEmirp(i) = n \wedge r \in Emirp]$$
$$\sqsubseteq \quad \langle\textbf{seq}\rangle$$
$$\llcorner count, r : [n > 0, Inv]\lrcorner_{(2)};$$
$$\llcorner count, r : [Inv, \sum_{i=2}^{r} isEmirp(i) = n \wedge r \in Emirp]\lrcorner_{(3)}$$

Where the loop invariant is defined by:

$$Inv = \left(\ count = \textstyle\sum_{i=2}^{r} isEmirp(i) \wedge n > 0 \wedge 0 \leq count \leq n \wedge r \geq count\ \right)$$

$(2) \sqsubseteq \quad \langle$Routine work: initialize the variables in the loop$\rangle$

$r := 1; count := 0;$

$(3) \sqsubseteq \quad \langle$**s-post**, justified below in 2.1$\rangle$

$count, r : [Inv, Inv \wedge (count = n \wedge r \in Emirp)]$

$\sqsubseteq \quad \langle$**while**, this simplification proof is above in 1.1.1$\rangle$

while $count \neq n \vee ISEMIRP(r) \neq 1$ **do**

$\llcorner count, r : [Inv \wedge (count \neq n \vee r \notin Emirp), Inv] \lrcorner_{(4)}$

od

$(4) \sqsubseteq \quad \langle$**s-post**, justified below in 2.2$\rangle$

$\llcorner count, r : [Inv \wedge (count \neq n \vee r \notin Emirp), Inv[^{count+isEmirp(r)}/_{count}][^{r+1}/_{r}]]; \lrcorner_{(5)}$

$(5) \sqsubseteq \quad \langle$**seq2**$\rangle$

$\llcorner r : [Inv \wedge (count \neq n \vee r \notin Emirp), Inv[^{r+1}/_{r}]]; \lrcorner_{(6)}$

$\llcorner count, r : [Inv[^{r+1}/_{r}], Inv[^{count+isEmirp(r)}/_{count}][^{r+1}/_{r}]] \lrcorner_{(7)}$

$(6) \sqsubseteq \quad \langle$**ass**$\rangle$

$r := r + 1;$

$(7) \sqsubseteq \quad \langle$**ass**$\rangle$

$count := count + ISEMIRP(r);$

We gather the code for the procedure body of EMIRP:

var $count$;

$r := 1;$

$count := 0;$

while $count \neq n \vee ISEMIRP(r) \neq 1$ **do**

$\quad r := r + 1;$

$\quad count := count + ISEMIRP(r);$

od

2.1 Proof of $pre(3)[^{count_0}/_{count}][^{r_0}/_{r}] \wedge (Inv \wedge (count = n \wedge r \in Emirp)) \Rightarrow post(3)$

$$
\begin{aligned}
& pre(3) \wedge Inv \wedge (count = n \wedge r \in Emirp) \\
\Leftrightarrow \quad & \langle \text{Expand } pre(3) \ \rangle \\
& Inv[^{count_0}/_{count}][^{r_0}/_{r}] \wedge (Inv \wedge (count = n \wedge r \in Emirp)) \\
\Leftrightarrow \quad & \langle \text{Substitute } Inv \text{ and } Inv[^{count_0}/_{count}][^{r_0}/_{r}] \ \rangle \\
& \left(\begin{array}{l} (count_0 = \sum_{i=2}^{r_0} isEmirp(i) \wedge n > 0 \wedge 0 \leq count_0 \leq n \wedge r_0 \geq count_0) \wedge \\ (count = \sum_{i=2}^{r} isEmirp(i) \wedge n > 0 \wedge 0 \leq count \leq n \wedge r \geq count \wedge \\ (count = n \wedge r \in Emirp)) \end{array} \right) \\
\Rightarrow \quad & \langle \text{Simplify and remove } count. \rangle \\
& \left(\begin{array}{l} (count_0 = \sum_{i=2}^{r_0} isEmirp(i) \wedge n > 0 \wedge 0 \leq count_0 \leq n \wedge r_0 \geq count_0) \wedge \\ (n = \sum_{i=2}^{r} isEmirp(i) \wedge r \geq n \wedge r \in Emirp) \end{array} \right) \\
\Rightarrow \quad & \langle \text{Directly imply from second line.} \rangle \\
& \sum_{i=2}^{r} isEmirp(i) = n \wedge r \in Emirp \\
\Leftrightarrow \quad & \langle \text{Definition of } post(3) \rangle \\
& post(3)
\end{aligned}
$$

2.2 Proof of $pre(4)[^{count_0}/_{count}][^{r_0}/_r] \wedge Inv[^{count+isEmirp(r)}/_{count}][^{r+1}/_r] \Rightarrow post(4)$

$$pre(4)[^{count_0}/_{count}][^{r_0}/_r] \wedge Inv[^{count+isEmirp(r)}/_{count}][^{r+1}/_r]$$

$\Leftrightarrow$ $\langle$Expand $pre(4)$ and substitute$\rangle$

$$(Inv \wedge (count_0 \neq n \vee r_0 \notin Emirp)) \wedge Inv[^{count+isEmirp(r)}/_{count}][^{r+1}/_r]$$

$\Leftrightarrow$ $\langle$Expand Inv and substitute$\rangle$

$$\left(\begin{array}{l} ((count_0 = \sum_{i=2}^{r_0} isEmirp(i) \wedge n > 0 \wedge 0 \leq count_0 \leq n \wedge r_0 \geq count_0) \\ (count_0 \neq n \vee r_0 \notin Emirp))\wedge \\ (count + isEmirp(r+1) = \sum_{i=2}^{r+1} isEmirp(i) \wedge n > 0 \wedge \\ 0 \leq count + isEmirp(r+1) \leq n \wedge r+1 \geq count + isEmirp(r+1)) \end{array}\right)$$

$\Rightarrow$ $\langle$Separate $\sum_{i=2}^{r+1} isEmirp(i)\rangle$

$$\left(\begin{array}{l} ((count_0 = \sum_{i=2}^{r_0} isEmirp(i) \wedge n > 0 \wedge 0 \leq count_0 \leq n \wedge r_0 \geq count_0) \\ (count_0 \neq n \vee r_0 \notin Emirp))\wedge \\ (count + isEmirp(r+1) = \sum_{i=2}^{r} isEmirp(i) + isEmirp(r+1) \wedge n > 0 \wedge \\ 0 \leq count + isEmirp(r+1) \leq n \wedge r+1 \geq count + isEmirp(r+1)) \end{array}\right)$$

$\Rightarrow$ $\langle$Merge $count_0$ and $count$, r_0 and $r\rangle$

$$\left(count = \sum_{i=2}^{r} isEmirp(i) \wedge n > 0 \wedge 0 \leq count \leq n \wedge r \geq count \right)$$

$\Leftrightarrow$ $\langle$Definition of Inv$\rangle$

$$Inv$$

$\Leftrightarrow$ $\langle$Definition of $post(4)\rangle$

$$post(4)$$

3 Task 3

4 Task 4