

Assignment 2

Ruofei HUANG(z5141448) Anqi ZHU(z5141541)

April 29, 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 multiplying two natural numbers (bigger than 1) smaller than itself¹. 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:

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

We shall use its procedure call sugar, **result** $p = ISPRIME(\mathbf{value} \ n)$, to verify primes later in our refinement.

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:

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

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

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

where:

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

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

$$\begin{aligned} &\mathbf{proc} \text{ reversen}(\mathbf{value} \ n : \mathbb{N}, \mathbf{result} \ v : \mathbb{N}). \\ &r, v : [\text{TRUE}, v = \text{rev}(r_0)] \end{aligned}$$

1.3 Emirp

An emirp n is a prime number that results in a different prime when its decimal digits are reversed³. The definition of emirp can be construct in mathematical semantics as follows:

$$n \in \text{Emirp} \iff n \in \text{Prime} \wedge \text{rev}(n) \in \text{Prime} \wedge n \neq \text{rev}(n)$$

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

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

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:

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

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.

$$\begin{aligned} &\mathbf{proc} \text{ ISEMIRP}(\mathbf{value} \ n : \mathbb{N}, \mathbf{result} \ w). \\ &\llbracket n, w : [\text{TRUE}, \left(\begin{aligned} &(w = 1 \wedge \text{rev}(n_0) \neq n_0 \wedge n_0 \in \text{Prime} \wedge \text{rev}(n_0) \in \text{Prime}) \vee \\ &(w = 0 \wedge \neg(\text{rev}(n_0) \neq n_0 \wedge n_0 \in \text{Prime} \wedge \text{rev}(n_0) \in \text{Prime})) \end{aligned} \right)] \rrbracket^{-(A)} \end{aligned}$$

³ Another reference from Wikipedia :<https://en.wikipedia.org/wiki/Emirp>

$$\begin{aligned}
(A) &\sqsubseteq \langle \mathbf{c\text{-}frame} \rangle \\
&w : [\text{TRUE}, \left(\begin{array}{l} (w = 1 \wedge \text{rev}(n) \neq n \wedge n \in \text{Prime} \wedge \text{rev}(n) \in \text{Prime}) \vee \\ (w = 0 \wedge \neg(\text{rev}(n) \neq n \wedge n \in \text{Prime} \wedge \text{rev}(n) \in \text{Prime})) \end{array} \right)] \\
&\sqsubseteq \langle \mathbf{i\text{-}loc} \rangle \\
&\mathbf{var} \ r \cdot r, w : [\text{TRUE}, \left(\begin{array}{l} (w = 1 \wedge \text{rev}(n) \neq n \wedge n \in \text{Prime} \wedge \text{rev}(n) \in \text{Prime}) \vee \\ (w = 0 \wedge \neg(\text{rev}(n) \neq n \wedge n \in \text{Prime} \wedge \text{rev}(n) \in \text{Prime})) \end{array} \right)] \\
&\sqsubseteq \langle \mathbf{seq2} \rangle \\
&\llbracket r : [\text{TRUE}, r = \text{rev}(n)]; \neg(B) \\
&\llbracket r \cdot w : [r = \text{rev}(n), \left(\begin{array}{l} (w = 1 \wedge \text{rev}(n) \neq n \wedge n \in \text{Prime} \wedge \\ \text{rev}(n) \in \text{Prime}) \vee (w = 0 \wedge \neg(\text{rev}(n) \neq n \\ \wedge n \in \text{Prime} \wedge \text{rev}(n) \in \text{Prime})) \end{array} \right)] \neg(C) \\
(B) &\sqsubseteq \langle \mathbf{proc} \rangle \\
&\text{reversen}(n, r); \\
(C) &\sqsubseteq \langle \mathbf{s\text{-}pos, justified below in 1.3.2} \rangle \\
&\llbracket r \cdot w : [r = \text{rev}(n), \left(\begin{array}{l} (w = 1 \wedge r \neq n \wedge n \in \text{Prime} \wedge r \in \text{Prime}) \vee \\ (w = 0 \wedge \neg(r \neq n \wedge n \in \text{Prime} \wedge r \in \text{Prime})) \end{array} \right)] \neg(C') \\
(C') &\sqsubseteq \langle \mathbf{if} \rangle \\
&\mathbf{if} \ r \neq n \\
&\mathbf{then} \ \llbracket r \cdot w : [r \neq n \wedge \text{pre}(C'), \text{post}(C')] \neg(D) \\
&\mathbf{else} \ \llbracket r \cdot w : [r = n \wedge \text{pre}(C'), \text{post}(C')] \neg(E) \\
&\mathbf{fi} \\
(D) &\sqsubseteq \langle \mathbf{if} \rangle \\
&\mathbf{if} \ \text{ISPRIME}(n) > 0 \\
&\mathbf{then} \ \llbracket r \cdot w : [\text{pre}(D) \wedge n \in \text{Prime}, \text{post}(C')] \neg(F) \\
&\mathbf{else} \ \llbracket r \cdot w : [\text{pre}(D) \wedge n \notin \text{Prime}, \text{post}(C')] \neg(G) \\
&\mathbf{fi} \\
(F) &\sqsubseteq \langle \mathbf{if} \rangle \\
&\mathbf{if} \ \text{ISPRIME}(r) > 0 \\
&\mathbf{then} \ \llbracket r \cdot w : [\text{pre}(F) \wedge r \in \text{Prime}, \text{post}(C')] \neg(H) \\
&\mathbf{else} \ \llbracket r \cdot w : [\text{pre}(F) \wedge r \notin \text{Prime}, \text{post}(C')] \neg(I) \\
&\mathbf{fi} \\
(E)(G)(I) &\sqsubseteq \langle \mathbf{ass, justified below in 1.3.3} \rangle \\
&w := 0 \\
(H) &\sqsubseteq \langle \mathbf{ass, justified below in 1.3.4} \rangle \\
&w := 1
\end{aligned}$$

We gather the code for the procedure body of ISEMIRP:

```

var  $r$ ;
 $reversen(n, r)$ ;
if  $r \neq n$  then
  if  $ISPRIME(n) > 0$  then
    if  $ISPRIME(r) > 0$ 
      then  $w := 1$ ;
      else  $w := 0$ ;
    fi
  else  $w := 0$ ;
  fi
else  $w := 0$ ;
fi

```

1.3.2 Proof of $pre(C)^{[r_0/r]}[^{w_0/w}] \wedge post(C') \Rightarrow post(C)$

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

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

1.4 Specification of the Main Procedure

The job of our main procedure, **proc** $EMIRP(\text{value } n : \mathbb{N}, \text{result } r)$, 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:

```

proc  $EMIRP(\text{value } n : \mathbb{N}, \text{result } r)$ .
 $\sqcup n, r : [n > 0, \sum_{i=2}^r isEmirp(i) = n_0 \wedge r \in Emirp] \sqcup_{(1)}$ 

```

2 Task 2

$$\begin{aligned}
(1) & \sqsubseteq \langle \mathbf{c-frame} \rangle \\
& r : [n > 0, \sum_{i=2}^r isEmirp(i) = n \wedge r \in Emirp] \\
& \sqsubseteq \langle \mathbf{i-loc} \rangle \\
& \mathbf{var} \ count \cdot count, r : [n > 0, \sum_{i=2}^r isEmirp(i) = n \wedge r \in Emirp] \\
& \sqsubseteq \langle \mathbf{seq} \rangle \\
& \sqsubseteq count, r : [n > 0, Inv] \dashv(2); \\
& \sqsubseteq count, r : [Inv, \sum_{i=2}^r isEmirp(i) = n \wedge r \in Emirp] \dashv(3)
\end{aligned}$$

where the loop invariant is defined by:

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

$$\begin{aligned}
(2) & \sqsubseteq \langle \text{Routine work: initialize the variables in the loop} \rangle \\
& r := 2; count := 0; \\
(3) & \sqsubseteq \langle \mathbf{s-post}, \text{justified below} \rangle \\
& count, r : [Inv, Inv \wedge (count = n \wedge r \in Prime)] \\
& \sqsubseteq \langle \mathbf{while} \rangle \\
& \mathbf{while} \ count \neq n \vee r \notin Prime \ \mathbf{do} \\
& \quad \sqsubseteq count, r : [Inv \wedge (count \neq n \vee r \notin Prime), Inv] \dashv(4) \\
& \mathbf{od} \\
(4) & \sqsubseteq \langle \mathbf{f-ass}, \text{justified below} \rangle \\
& \sqsubseteq count, r : [Inv \wedge (count \neq n \vee r \notin Prime), Inv^{r+1}/r] \dashv(5) \\
& r := r + 1; \\
(5) & \sqsubseteq \langle \mathbf{c-frame}, \mathbf{ass}, \text{justified below} \rangle \\
& count := count + ISEMIRP(r);
\end{aligned}$$

We gather the code for the procedure body of EMIRP:

```
var count;
r := 2;
count := 0;
while count  $\neq$  n  $\vee$  r  $\notin$  Prime do
    r := r + 1;
    count := count + ISEMIRP(r);
od
```

2.1 Task 3

2.2 Task 4