

# Assignment 2

Ruofei HUANG(z5141448)      Anqi ZHU(z5141541)

April 28, 2018

## 1 Task 1

### 1.1 Prime

A number is a prime is a natural number greater than 1 that cannot be formed by multiplying two smaller number<sup>1</sup>. Hence we can have the definition by set theory:

$$Prime = \{n | \neg \exists x \in (2..n-1) (x|n)\}$$

Then we can base on the GMP function that make up the spec of the procedure call of **proc** *ISPRIME*(**value** *n*, **result** *r*)

**proc** *ISPRIME*(**value** *n*, **result** *r*).  
[*TRUE*, (*n* ∈ *Prime* ∧ *r* > 0) ∨ (*n* ∉ *Prime* ∧ *r* = 0)]

Which means  $ISPRIME(n) > 0^2 \Rightarrow n \in Prime$ .

### 1.2 Reverse

To define a emirp, a mathematical function to describe a number's reverse would be helpful. By the spec in verifying reversen<sup>3</sup>, we have this definition:

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

---

<sup>1</sup> Direct reference from Wikipedia: [https://en.wikipedia.org/wiki/Prime\\_number](https://en.wikipedia.org/wiki/Prime_number)

<sup>2</sup> This act as another procedure call sugar  $r := ISPRIME(n)$ ; the complete expanded version will be  $ISPRIME(n, r); ((r > 0 \Rightarrow r \in Prime) \vee (\text{else} \Rightarrow r \notin Prime))$ . The continuous prove will follow this conversion to make life easier.

<sup>3</sup> A proof provided by Lecturer in Control of this course on <https://www.cse.unsw.edu.au/~cs2111/18s1/lec/reverse.pdf>

where:

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

Hence, we can justify the spec of **proc** *reversen*(**value**  $n : \mathbb{Z}$ , **result**  $r : \mathbb{Z}$ ) as

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

### 1.3 Emirp

An emirp is a prime number that results in a different prime when its decimal digits are reverse<sup>4</sup>. Hence a definition of emirp can be construct as follow:

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

We also construct another function to help us find the  $n^{\text{th}}$  emirp, which is as follow:

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

#### 1.3.1 Procedure Call

Also for our usage in the procedure call in the main programme, we have develop a procedure to do the same thing. Hence we have this spec

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

#### 1.3.2 Refinement Calculation

$$\begin{aligned} (A) &\sqsubseteq \langle \mathbf{c-frame} \rangle \\ &w : [\text{TRUE}, \left( \begin{array}{l} (w = 1 \wedge \text{rev}(n) \neq n \wedge n \in \text{Prime} \wedge \text{rev}(n) \in \text{Prime}) \vee \\ (w \neq 1 \wedge \neg(\text{rev}(n) \neq n \wedge n \in \text{Prime} \wedge \text{rev}(n) \in \text{Prime})) \end{array} \right) ] \end{aligned}$$

---

<sup>4</sup> Another reference from Wikipedia :<https://en.wikipedia.org/wiki/Emirp>

$$\begin{aligned}
& \sqsubseteq \quad \langle \mathbf{i\text{-}loc} \rangle \\
& \quad r \cdot 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 \neq 1 \wedge \neg(\text{rev}(n) \neq n \wedge n \in \text{Prime} \wedge \text{rev}(n) \in \text{Prime})) \end{array} \right)] \\
& \sqsubseteq \quad \langle \mathbf{seq2} \rangle \\
& \quad \llcorner r : [\text{TRUE}, r = \text{rev}(n)]; \lrcorner(B) \\
& \quad \llcorner 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 \neq 1 \wedge \neg(\text{rev}(n) \neq n \wedge n \in \text{Prime} \wedge \text{rev}(n) \in \text{Prime})) \end{array} \right)] \lrcorner(C') \\
(B) & \sqsubseteq \quad \langle \mathbf{ass, need to be justify?} \rangle \\
& \quad r := \text{reversen}(n) \\
(C) & \sqsubseteq \quad \langle \mathbf{s\text{-}pos}, r = \text{rev}(n), \text{replace all the rev in the post} \rangle \\
& \quad \llcorner 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 \neq 1 \wedge \neg(r \neq n \wedge n \in \text{Prime} \wedge r \in \text{Prime})) \end{array} \right)] \lrcorner(C') \\
& \sqsubseteq \quad \langle \mathbf{if} \rangle \\
& \quad \mathbf{if} \ r \neq n \\
& \quad \mathbf{then} \llcorner r \cdot w : [r \neq n, \text{post}(C')] \lrcorner(D) \\
& \quad \mathbf{else} \llcorner r \cdot w : [r = n, \text{post}(C')] \lrcorner(E) \\
& \quad \mathbf{fi} \\
(D) & \sqsubseteq \quad \langle \mathbf{if} \rangle \\
& \quad \mathbf{if} \ \text{ISPRIME}(n) > 0 \\
& \quad \mathbf{then} \llcorner r \cdot w : [r \neq n \wedge n \in \text{Prime}, \text{post}(C')] \lrcorner(F) \\
& \quad \mathbf{else} \llcorner r \cdot w : [r \neq n \wedge n \notin \text{Prime}, \text{post}(C')] \lrcorner(G) \\
& \quad \mathbf{fi} \\
(F) & \sqsubseteq \quad \langle \mathbf{if} \rangle \\
& \quad \mathbf{if} \ \text{ISPRIME}(r) > 0 \\
& \quad \mathbf{then} \llcorner r \cdot w : [r \neq n \wedge n \in \text{Prime} \wedge r \in \text{Prime}, \text{post}(C')] \lrcorner(G) \\
& \quad \mathbf{else} \llcorner r \cdot w : [r \neq n \wedge n \in \text{Prime} \wedge r \notin \text{Prime}, \text{post}(C')] \lrcorner(H) \\
& \quad \mathbf{fi} \\
(E)(G)(H) & \sqsubseteq \quad \langle \mathbf{ass, need to be justify} \rangle \\
& \quad w := 0 \\
(G) & \sqsubseteq \quad \langle \mathbf{ass, need to be justify} \rangle \\
& \quad w := 1
\end{aligned}$$

### 1.3.3 Justification of

### 1.3.4 Toy Language Code

```
 $r := \text{reversen}(n)$  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
```

## 1.4 Pre- and Postcondition

Our task is find the  $n^{th}$  emrip, by the previous definition of  $isEmirp(n)$  we can construct the pre- and postcondition in this way:

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

## 2 Task 2