Assignment 3 - Emirps

Rahil Agrawal z5165505
Aditya Karia z5163287

COMP2111 18s1

1 Task 1 -Syntactic Data Type Dict

We define a syntactic data type **Dict** that encapsulates a dictionary set W as follows :

$$\textbf{Dict} = (init^{Dict}, (W, x : [pre_j^{Dict}, post_j^{Dict}]_{j \in J}))$$

which consists of an initialisation predicate $init^{Dict} = (W = \{\})$ and the following operations:

$proc\ addword^{Dict}$(value w) . $W : [True, W = W_0 \cup w]$
$proc\ checkword^{Dict}$(value W, value w, result b) . $b : [True, b = (W = \{\})]$
$proc\ delword^{Dict}$(value w) . $W : [W \neq \{\}, W = W_0 \setminus w]$

2 Task 2 - Refinement to DictA

Were refining this to a data type DictA where we replace W with a Trie t. From Ass3 2018 S1 Specification, "A *trie domain* is a prefixclosed finite subset of L^*. A *trie* is a function from a trie domain to Booleans. Given a trie t we write **dom**t for its trie domain. Let T be the set of all tries."

The correspondance between the two data types is captured by the function $f : T->$ $P(L^*)$ given by :

$$f(t) = \{w \in L^* \mid w \in \textbf{dom}t \wedge t(w) = 1\}$$

Also, from Ass3 2018 S1 Specification, "Formally, we write $v \leq w$ if word $v \in L^*$ is a prefix of $w \in L^*$, i.e., $\exists v' \in L^* (vv' = w)$. We write B for 0, 1 where 0 represents false and 1 true."

We define our With the aforementioned facts in mind, we define a concrete syntactic data type **DictA** that encapsulates a trie t as follows :

I

$$\mathbf{DictA} = (init^{DictA}, (t, x : [pre_j^{DictA}, post_j^{DictA}]_{j \in J}))$$

which consists of an initialisation predicate $init^{DictA} = (t = \{\})$ and the following operations:

$proc\ addword^{DictA}(\text{value w}) \ . \ t : [True, post(addword^{DictA})]$ where

$$post(addword^{DictA}) = \mathbf{dom}\,t = \mathbf{dom}\,t_0 \cup \{v \in L^* \mid v \leq w\} \wedge t = t_0 \cup$$
$$\{v \in L^* \mid v \in \mathbf{dom}\,t \ \wedge v < w \wedge t_0(v) = 1 \Rightarrow t(v) = 1$$
$$v < w \wedge t_0(v) = 0 \Rightarrow t(v) = 0$$
$$v = w \Rightarrow t(v) = 1\}$$

$proc\ checkword^{DictA}(\text{value t, value w, result b}) \ . \ b : [True, b = (t(w) = 1)]$
$proc\ delword^{DictA}(\text{value w}) \ . \ t : [t \neq \{\}, t(w) = 0]$

That this indeed is a refinement requires checking the relevant proof obligations:

$$init^{DictA} \Rightarrow init^{Dict}[f(t)/_W] \tag{1}$$
$$pre_j^{Dict}[f(t)/_W] \Rightarrow pre_j^{DictA}, for\ j \in J \tag{2}$$
$$pre_j^{Dict}[f(t_0),x_0/_{W,x}] \wedge post_j^{DictA} \Rightarrow post_j^{Dict}[f(t_0),f(t)/_{W_0,W}], for\ j \in J \tag{3}$$

We begin with (1).

$$init^{DictA}[f(t_0),x_0/_{W,x}] = f(t) \neq \{\}$$
$$\Rightarrow \langle \text{def. of f} \rangle$$
$$f(t) = \{\} \Rightarrow$$
$$\Rightarrow \langle \text{def. of Dict} \rangle$$
$$init^{Dict}[f(t)/_W]$$

Condition (2) is only required to be proven when the concrete pre-condition is non-trivial (not True). This is only the case in delword.

$$pre_{delword}^{Dict}[f(t_0),x_0/_{W,x}] = f(t) \neq \{\}$$
$$\Rightarrow \langle \text{def. of f} \rangle$$
$$w \in \mathbf{dom}\,t \wedge t(w) = 1$$
$$\Rightarrow \langle \text{def. of trie} \rangle$$
$$t \neq \{\}$$
$$\Rightarrow \langle \text{def. of } pre_{delword}^{DictA} \rangle$$
$$pre_{delword}^{DictA}$$

Finally, condition (4) needs to be checked for all operations.
For addword, we prove

$$LHS = BLAH$$
$$\Rightarrow \langle BLAH \rangle$$
$$BLAH$$
$$\Rightarrow \langle BLAH \rangle$$
$$BLAH$$
$$\Rightarrow \langle BLAH \rangle$$
$$RHS$$

For checkword, we prove

$$LHS = BLAH$$
$$\Rightarrow \langle BLAH \rangle$$
$$BLAH$$
$$\Rightarrow \langle BLAH \rangle$$
$$BLAH$$
$$\Rightarrow \langle BLAH \rangle$$
$$RHS$$

For delword, we prove

$$LHS = BLAH$$
$$\Rightarrow \langle BLAH \rangle$$
$$BLAH$$
$$\Rightarrow \langle BLAH \rangle$$
$$BLAH$$
$$\Rightarrow \langle BLAH \rangle$$
$$RHS$$

3 Task 3 - Derivation

$$\textbf{proc } FunctionName(\textbf{value } n, \textbf{result } r) \cdot \llcorner n, r, x : \begin{bmatrix} Pre, Post \end{bmatrix} \lrcorner_{(1)}$$

$(1) \sqsubseteq \quad \langle \textbf{Rule} \rangle$

$$\llcorner r, x : \begin{bmatrix} NewPre, NewPost \end{bmatrix} \lrcorner_{(2)}$$

$(2) \sqsubseteq \quad \langle \textbf{seq} \rangle$

$$\llcorner i, x, r : \begin{bmatrix} Pre, Blah \end{bmatrix} \lrcorner_{(3)};$$

$$\llcorner i, x : \begin{bmatrix} Blah, Post \end{bmatrix} \lrcorner_{(4)}$$

$\sqsubseteq \quad \langle \textbf{ass - (1)} \rangle$

$\textbf{i} := \textbf{1}$

$\textbf{x} := \textbf{13}$

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

$$\llcorner s, x : \begin{bmatrix} pre(16), s = 0 \wedge x > 0 \end{bmatrix} \lrcorner_{(18)};$$

$$\llcorner a, i, r, s, x : \begin{bmatrix} x > 0 \wedge s = 0, post(16) \end{bmatrix} \lrcorner_{(19)}$$

$(5) \sqsubseteq \quad \langle \textbf{while} \rangle$

$\textbf{while } \textbf{j} \neq \textbf{r} \textbf{ do}$

$$\llcorner r, j : \begin{bmatrix} Inv_2 \wedge j \neq r, Inv_2 \end{bmatrix} \lrcorner_{(8)}$$

$\textbf{od};$

$(9) \sqsubseteq \quad \langle \textbf{if} \rangle$

$\textbf{if } Gaurd$

$\textbf{then } \llcorner a : [Gaurd \wedge pre(9), post(9)]\lrcorner_{(11)}$

$\textbf{else } \llcorner a : [Not \ Gaurd \wedge pre(9), post(9)]\lrcorner_{(12)}$

$\textbf{fi};$

We gather the code for the procedure body of blah:

blah(r, n) :
    $var\ i := 1;$
    $var\ x := 13;$
    $r := 13;$
    while $j \neq r$ **do**
        $x := x + 1;$
        $var\ a := 1;$
        $isPrime(x, a);$
        if $a = 1$ **then**
            $var\ s := 0;$
            $reversen(x, s);$
            $var\ b := 1;$
            $isPrime(s, b);$
            if $b = 1 \wedge s \neq x$ **then**
                $i := i + 1;$
                $r := x;$
    od;

Also, we gather the code for the procedure body of blah:

isPrime(r, j) :
    $var\ j := 2;$
    while $j \neq r$ **do**
        if $(r \bmod j) = 0$ **then**
            $a := 0;$
        $j := j + 1;$
    od;

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

3.1 Implication 1: $[BLAH, BLAH] \sqsubseteq BLAH$

To prove:$BLAH \Rightarrow BLAH$

Proof:
$LHS = BLAH$
$\Rightarrow \langle \text{BLAH} \rangle$
$BLAH$
$\Rightarrow \langle \text{BLAH} \rangle$
$BLAH$
$\Rightarrow \langle \text{BLAH} \rangle$
$BLAH$
$\Rightarrow \langle \text{BLAH} \rangle$
$BLAH$
$\Rightarrow \langle \text{BLAH} \rangle$
RHS

3.2 Implication 2: $[BLAH, BLAH] \sqsubseteq BLAH$

To prove:$BLAH \Rightarrow BLAH$

Proof:
$LHS = BLAH$
$\Rightarrow \langle \text{BLAH} \rangle$
$BLAH$
$\Rightarrow \langle \text{BLAH} \rangle$
$BLAH$
$\Rightarrow \langle \text{BLAH} \rangle$
$BLAH$
$\Rightarrow \langle \text{BLAH} \rangle$
$BLAH$
$\Rightarrow \langle \text{BLAH} \rangle$
RHS

4 Task 4 - C Code

```
1  #include "dict.h"
2  #include <stdio.h>
```