Assignment 3

Ruofei HUANG(z5141448) Anqi ZHU(z5141541)

May 28, 2018

1 Task 1

1.1 Type *word*

This definiation of wrod is basicly from requirement of assignment 3 We say two words v, w that v is an absolute prefix of w as $v < w$ which is define as $v \leq w \wedge v \neq w$.

1.2 abstract Data Type *Dict*

According to the specified problem statement in the assignment, we could describe the syntactic data type *Dict* as below. The encapsulated state is a dictionary word set W.

$$Dict = (W = \phi,$$
$$\left(\begin{array}{l} \textbf{proc } addword^{Dict}(\textbf{word } w) \cdot b, W : [\text{TRUE}, b = b_0 \wedge W = W_0 \cup \{w\}] \\ \textbf{func } checkword^{Dict}(\textbf{word } w) : \mathbb{B} \cdot \textbf{var } b \cdot \ b, W : [\text{TRUE}, b = (w \in W_0)];\ \textbf{return } b \\ \textbf{proc } delword^{Dict}(\textbf{word } w) \cdot b, W : [w \in W, b = b_0 \wedge W = W_0 \backslash \{w\}] \end{array}\right))$$

2 Task 2

Now we would like to refine *Dict* to a second data type *DictA* where we replace W with a trie t,the corresponding trie domain $D = \textbf{dom}(t)$. It represents the set of all tries according to the domain. We shall use this definition later in our refinement.

2.1 Datat Invariant

$$\forall w \in \textbf{dom}(t), t(w) = 1, w' \leq w\,(w' \in \textbf{dom}(t))$$

2.2 Data Type Refinement

This suggests we should first build up a inductively defined predicate to ensure the provable relations between $DictA$ and $Dict$.

$$r = (W = \{w \in \mathbf{dom}(t) | t(w) = 1\})$$

which we can translate into a function from concrete to abstract values:

$$f(t) = \{w \in \mathbf{dom}(t) | t(w) = 1\}$$

With that in mind we propose the initialisation predicate $init^{DictA} = (\mathbf{dom}(t) = \{\epsilon\} \wedge f(t) = \phi)$ and operations given as follows.

$$\begin{array}{l}
\textbf{proc } addword^{DictA}(\textbf{word } w) \cdot b, t : \\
\quad [\ \text{TRUE}, b = b_0 \wedge t = t_0 \cup \{w \mapsto 1\} \cup \{w' < w \wedge w' \notin \mathbf{dom}(t) | w' \mapsto 0\}] \\
\textbf{func } checkword^{DictA}(\textbf{word } w)\mathbb{B} \cdot \textbf{var } b \cdot b, t : \\
\quad [\text{TRUE}, b = (w \in \mathbf{dom}(t) \wedge t = t_0)];\ \textbf{return } b \\
\textbf{proc } delword^{DictA}(\textbf{word } w) \cdot b, t : \\
\quad [\text{TRUE}, b = b_0 \wedge (w \notin \mathbf{dom}(t) \vee t := t : w \mapsto 0)]
\end{array}$$

2.3 Proof of Refinement

We need start the proof with the initialisation:

$$\begin{array}{ll}
& init^{DictA} \Rightarrow init^{Dict}[^{f(t)}/_{W}] \\
\Leftrightarrow & \quad \langle\text{Definition of } init^{DictA} \text{ and } init^{Dict}\rangle \\
& \forall w \in \mathbf{dom}(t)\ (t(w) = 0) \Rightarrow W = \phi
\end{array}$$

Since all our precondition of concrete is trivial which all of them are TRUE, we don't need to proof the condition (3_f).But condition (4_f) must be checked for all three operations. For the *addword* we proof:

$$\begin{array}{ll}
& pre_{addword^{Dict}}[^{f(t_0)}/_{W}] \wedge post_{addword^{DictA}} \\
\Leftrightarrow & \quad \langle\text{Definition of } addword^{Dict} \text{ and } addword^{DictA}\ \rangle \\
& \text{TRUE}[^{f(t_0)}/_{W}] \wedge b = b_0 \wedge t = t_0 \cup \{w \mapsto 1\} \cup \{w' < w \wedge w' \notin \mathbf{dom}(t) | w' \mapsto 0\} \\
\Rightarrow & \quad \langle\text{Definition of } f\ \rangle \\
& f(t) = f(t_0 \cup \{w \mapsto 1\} \cup \{w' < w \wedge w' \notin \mathbf{dom}(t) | w' \mapsto 0\}) \\
\Leftrightarrow & \quad \langle\text{Logic}\rangle \\
& f(t) = f(t_0) \cup \{w\} \\
\Leftrightarrow & \quad \langle\text{Definition of } addword^{Dict} \text{ and } addword^{DictA}\ \rangle \\
& post_{addword^{Dict}}[^{f(t_0), f(t)}/_{W_0, W}]
\end{array}$$

For the *checkword* we proof:

$$
\begin{aligned}
& pre_{checkword^{Dict}}[^{f(t_0)}/_W] \wedge post_{checkword^{DictA}} \\
\Leftrightarrow \quad & \langle \text{Definition of } checkword^{DictA} \text{ and } checkword^{Dict} \rangle \\
& \text{TRUE}[^{f(t_0)}/_W] \wedge b = (w \in \mathbf{dom}(t) \wedge t = t_0) \\
\Rightarrow \quad & \langle \text{Definition of } f \rangle \\
& b = (w \in f(t)) \wedge t = t_0 \\
\Leftrightarrow \quad & \langle \text{Definition of } checkword^{DictA} \text{ and } checkword^{Dict} \rangle \\
& post_{checkword^{Dict}}[^{f(t_0), f(t)}/_{W_0, W}]
\end{aligned}
$$

For the *delword* we proof:

$$
\begin{aligned}
& pre_{delword^{Dict}}[^{f(t_0)}/_W] \wedge post_{delword^{DictA}} \\
\Leftrightarrow \quad & \langle \text{Definition of } delword^{DictA} \text{ and } delword^{Dict} \rangle \\
& w \in f(t_0) \wedge b = b_0 \wedge (w \notin \mathbf{dom}(t) \vee t := t : w \mapsto 0) \\
\Rightarrow \quad & \langle \text{Definition of } f \rangle \\
& f(t) = f(t_0) \backslash \{w\} \\
\Leftrightarrow \quad & \langle \text{Definition of } delword^{DictA} \text{ and } delword^{Dict} \rangle \\
& post_{delword^{Dict}}[^{f(t_0), f(t)}/_{W_0, W}]
\end{aligned}
$$

3 Task 3

We derive our code in to fours part by *init* and its operations.

3.1 init

From the spec we have:

$$
\begin{aligned}
& \mathbf{dom}(t) = \{\epsilon\} \wedge f(t) = \phi \\
\sqsubseteq \quad & \langle \text{ass} \rangle \\
& t := \{\epsilon \mapsto 0\}
\end{aligned}
$$

3.2 addword

From the spec we have:

$$
\begin{aligned}
& \mathbf{proc}\ addword^{DictA}(\mathbf{word}\ w) \cdot \\
& \quad \llcorner b, t : [\text{TRUE}, b = b_0 \wedge t = t_0 \cup \{w \mapsto 1\} \cup \{w' < w \wedge w' \notin \mathbf{dom}(t) | w' \mapsto 0\}] \lrcorner_{(A1)}
\end{aligned}
$$

$$
\begin{aligned}
(A1) \sqsubseteq \quad & \langle \text{c-frame} \rangle \\
& t : [\text{TRUE}, t = t_0 \cup \{w \mapsto 1\} \cup \{w' < w \wedge w' \notin \mathbf{dom}(t) | w' \mapsto 0\}]
\end{aligned}
$$

3.3 checkword

From the spec we have:

$$\textbf{func } checkword^{DictA}(\textbf{word } w)$$
$$\llcorner \mathbb{B} \cdot \textbf{var } b \cdot b, t : [\text{TRUE}, b = (w \in \mathbf{dom}(t) \wedge t = t_0)];\ \textbf{return } b \lrcorner_{(C1)}$$

$$(C1) \sqsubseteq \quad \langle \text{c-frame} \rangle$$
$$\mathbb{B} \cdot \textbf{var } b \cdot b : [\text{TRUE}, b = (w \in \mathbf{dom}(t))];\ \textbf{return } b$$
$$\sqsubseteq \quad \langle \text{proc, } 0 \leq |w|\ \rangle$$
$$\textbf{return } doCheckword(w, 0);$$

Where we define a recursive procedure call to do the dirty work also align the pre and post condition:

$$\textbf{func } doCheckword(\textbf{word } w, \textbf{var } index : \mathbb{N})$$
$$\llcorner \mathbb{B} \cdot \textbf{var } b \cdot b, \textbf{var } index[prefix \leq w, b = (w \in \mathbf{dom}(t))];\ \textbf{return } b \lrcorner_{(C2)}$$

$$(C2) \sqsubseteq \quad \langle \text{if} \rangle$$
$$\textbf{if } index < |w|$$
$$\textbf{then} \llcorner b, indexp[index < |w| \wedge pre(C2), post(C2)] \lrcorner_{(C3-1)}$$
$$\textbf{else} \llcorner b, index[index < |w| \wedge pre(C2), post(C2)] \lrcorner_{(C3-2)}$$
$$\textbf{fi}$$
$$(C3-1) \sqsubseteq \quad \langle \text{func} \rangle$$
$$\textbf{return } doCheckword(w, index + 1);$$

3.4 delword

From the spec we have:

$$\textbf{proc } delword^{DictA}(\textbf{word } w)$$
$$\llcorner \cdot b, t : [\text{TRUE}, b = b_0 \wedge (w \notin \mathbf{dom}(t) \vee t := t : w \mapsto 0)] \lrcorner_{(D1)}$$