Derivation of breath first search

COMP2111 assignment 3

Dae Ro Lee z5060887 and Wing Feng z5091907

May 31, 2017

1 Introduction

The derivation of BREATH FIRST SEARCH on a tree using a bounded QUEUE.
Firstly we extend the toy language with queues and queue operations, and operations on abstract queues, which we eliminate later with data refinement.

$$\begin{array}{lll} Q(\tau) ::= \langle\rangle & & \text{(empty queue)} \\ \mid \quad \langle\ x, t\ \rangle & & \text{where } x \in \tau \text{ and } t \in\ Q(\tau) \\ \mid \quad \langle\ t, x\ \rangle & & \end{array}$$

We define the following queue operation:
initialise: initialise a queue that can hold up to N elements to the empty queue value.

$$\text{INITIALISE} \qquad q : [\text{TRUE}, q = \langle\rangle] \sqsubseteq q := \langle\rangle$$

enqueue: adds an item to a queue if there's a space available

$$\text{ENQUEUE} \qquad q : [\text{TRUE}, q = \langle q_0, x\rangle] \sqsubseteq \text{ENQUEUE}(q, x)$$

dequeue: return the oldest item in the queue and remove it form the queue

$$\text{DEQUEUE} \qquad q, x : [q =< a, b >, x = a \land q = b] \sqsubseteq x := \text{DEQUEUE}(q)$$

2 The Derivation

proc SEARCH(**value** t, **value** N, **value** k, **result** v, **result** f) $\cdot$

$$\llcorner t, N, k, v, f : \left[\begin{array}{l} \forall x \in V_t(x \in \Gamma_t^*(r_t) \land x \notin \Gamma_t^+(x)) \land \max_{i\in\mathbb{N}} |\Gamma_t^i(r_t) \cup \Gamma_t^{i+1}(r_t)| \leq N, \\ (f \land \exists w \in V_{t_0}(k_{t_0}(w) = k_0 \land \lambda_{t_0}(w) = v)) \lor \\ (\neg f \land \forall w \in V_{t_0}(k_{t_0}(w) \neq\ k_0)) \end{array} \right] \lrcorner(1)$$

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

$$v, f : \left[\begin{array}{l} \forall x \in V_t(x \in \Gamma_t^*(r_t) \wedge x \notin \Gamma_t^+(x)) \wedge \max_{i \in \mathbb{N}} |\Gamma_t^i(r_t) \cup \Gamma_t^{i+1}(r_t)| \leq N, \\ (f \wedge \exists w \in V_t(k_t(w) = k \wedge \lambda_t(w) = v)) \vee \\ (\neg f \wedge \forall w \in V_t(k_t(w) \neq \ k)) \end{array} \right]$$

$$\sqsubseteq \quad \langle \textbf{introduce local variable} \rangle$$

$$\textbf{var } q, n \cdot \llcorner q, n, v, f : \left[\begin{array}{l} \forall x \in V_t(x \in \Gamma_t^*(r_t) \wedge x \notin \Gamma_t^+(x)) \wedge \max_{i \in \mathbb{N}} |\Gamma_t^i(r_t) \cup \Gamma_t^{i+1}(r_t)| \leq N, \\ (f \wedge \exists w \in V_t(k_t(w) = k \wedge \lambda_t(w) = v)) \vee \\ (\neg f \wedge \forall w \in \ V_t(k_t(w) \neq \ k)) \end{array} \right] \lrcorner(2)$$

3 Invariant

We define the loop invariant for BREATH FIRST SEARCH as:

$$I = \left(\begin{array}{l} \exists j \in \mathbb{N} \cdot (\forall \ i < j \cdot \forall \ x \in \Gamma^i(r_t) \cdot K_t(x) \neq k) \\ \wedge 0 \leq \ n \leq \ N \wedge \exists \ l \in [0, n-1] \cdot q[0..l] \subseteq \Gamma_t^j(r_t) \\ \wedge q[l+1..n-1] \subseteq \Gamma_t^{j+1}(r_t) \\ \wedge (\forall y \in V_t(y \in \Gamma_t^*(r_t) \wedge y \notin \Gamma_t^+(y))) \\ \wedge ((f \wedge \exists \ s \in \Gamma_t^j(r_t) \cdot k_t(s) = k \wedge \lambda_t(s) = v \wedge s \notin q) \\ \vee (\neg \ f \wedge \forall \ p \in \Gamma_t^j(r_t) \wedge p \notin \ q \wedge k_t(p) \neq k \wedge \Gamma_t(p) \subseteq q)) \end{array} \right)$$

j is defined as a node's distance from the root node, or the level of that node in the tree, hence at root node, j = 0.
The first line of the invariant shows that all previous levels of the tree before the current level has been visited and the key was not found.
the second line and third line of the invariant show that the queue, q is composed of at most only two levels of the tree at any given time, j (being defined at the current level) and $j + 1$.
The fifth line of the invariant says that if f then for some node on the current level there exists the node we are looking for.
The sixth line says all current level nodes that are not in q, are not the nodes we are looking for and their children are a subset of q.

Since $0 \leq \ n \leq \ N$, and n is described as the length of the queue
$\max_{i \in \mathbb{N}} |\Gamma_t^i(r_t) \cup \Gamma_t^{i+1}(r_t)| \leq N$, of the precondition implies the second and third line of the invariant. Showing that n the length of the queue cannot exceed the number of nodes of two consecutive levels of the tree, hence $n \leq \ N$, N representing the number of nodes of the last two levels of the tree.
The invariant also contains the fact that all nodes are connected as a tree data structure, identical to that in the precondition.

$I \wedge (f \vee\ n = 0)$, When $f = \text{TRUE}$ the invariant describes that some node,
$s \in \Gamma_t^j(r_t) \cdot k_t(s) = k \wedge \lambda_t(s) = v \wedge s \notin q$, which implies the post condition in the scenario that $f = \text{TRUE}$
In the case that $f = \text{FALSE} \wedge\ n = 0$ the first and last line of the invariant shows that all nodes previous to current level have been checked.
Since the queue is empty all nodes on the current level have also been checked implying that none of the nodes in the tree contain the key that we are looking for,
implying the post condition in the scenario that $n = 0$.
Therefore $I \wedge (f \vee\ n = 0)$ implies the post condition

4 The Derivation continued ...

$$
\begin{array}{ll}
(2) \sqsubseteq & \langle \textbf{seq} \text{ no initial variables} \rangle \\
& \llcorner q, n, v, f : [pre(2), I] \lrcorner_{(3)} \\
& ; \llcorner q, n, v, f : [I, post(2)] \lrcorner_{(4)}
\end{array}
$$

$$
\begin{array}{ll}
(3) \sqsubseteq & \langle \textbf{seq} \text{ no initial variables} \rangle \\
& \llcorner q, n, v, f : [pre(3), pre(3) \wedge q = <> \wedge \neg f] \lrcorner_{(5)} \\
& ; \llcorner q, n, v, f : [pre(3) \wedge q = <> \wedge \neg f, I] \lrcorner_{(6)}
\end{array}
$$

We derive code by initialising f to FALSE and ENQUEUE the root node to establish our invariant initially.

$$
\begin{array}{ll}
(5) \sqsubseteq & \langle \textbf{ass} \rangle \\
& q := initialise(); \\
& f := \text{FALSE}
\end{array}
$$

$$
\begin{array}{ll}
(6) \sqsubseteq & \langle \textbf{ass} \rangle \\
& \text{ENQUEUE}(q, r_t); \\
& n := 1
\end{array}
$$

$$
\begin{array}{ll}
(4) \sqsubseteq & \langle \textbf{s-post} \text{ justified above in Sect. 3} \rangle \\
& q, n, v, f : [I \wedge\ g, I \wedge (f \vee\ q = <>)] \\
\sqsubseteq & \langle \textbf{while} \rangle \\
& \textbf{while } n \neq 0 \wedge \neg f \textbf{ do}
\end{array}
$$

$\llcorner q, n, v, f : [I \wedge\ n \neq\ 0 \wedge \neg\ f, I] \lrcorner_{(7)}$

od

$(7) \sqsubseteq$ $\langle$**i-loc**$\rangle$

var $tmp \cdot \llcorner tmp, q, n, v, f : [I \wedge\ n \neq\ 0 \wedge \neg\ f, I] \lrcorner_{(8)}$

$(8) \sqsubseteq$ $\langle$**seq** no initial variables$\rangle$

$\llcorner tmp, q, n, v, f : [n \neq\ 0 \wedge \neg\ f \wedge\ I \wedge\ q =< z, qt >, \neg\ f \wedge\ q = qt \wedge tmp = z] \lrcorner_{(a)}$

$; \llcorner tmp, q, n, v, f : [q = qt \wedge tmp = z, I] \lrcorner_{(b)}$

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

if $k_t(tmp) = k$ **then**

$\llcorner tmp, q, n, v, f : [\neg\ f \wedge k_t(tmp) = k, I] \lrcorner_{(c)}$

else

$\llcorner tmp, q, n, v, f : [k_t(tmp) \neq k, I] \lrcorner_{(d)}$

fi

$(a) \sqsubseteq$ $\langle$**dequeue**$\rangle$

$tmp :=$ DEQUEUE(q, n)

To make the invariant TRUE if $k_t(tmp) = k$, we have to assign $f :=$ TRUE because the sixth line of the invariant tell us that f cannot be FALSE when $k_t(tmp) = k$.
Now make the fifth line of the invariant true assign
$\lambda_t(tmp) = v$ such that the invariant is $True$.

$(c) \sqsubseteq$ $\langle$**ass**$\rangle$

$f :=$ TRUE;
$v = \lambda_t(tmp)$

Because tmp is not the node we are looking for, we now have to make sure the conditions of the invariant are met. The invariant tells us that q contains up-to two levels of the tree at any given time. We also know that since tmp has been DEQUEUE from q, $tmp \notin q$, hence the end of the 5th line of the invariant tells us that the children of all the nodes that are on the current level but not in q must be a subset of q, therefore all the children of tmp have to be added to q to meet the invariant.
We do this by looping through the set of successors and ENQUEUE them to q.

$(d) \sqsubseteq$ $\langle$**...**$\rangle$

var $m := \Gamma_t(tmp)$;
while $m \neq \emptyset$ **do**
var $c :\in\ m$;

$\text{ENQUEUE}(q, c);$
$n := n + 1;$
$m := m \backslash \{c\};$
od

5 Coupling Invariant

We do the data refinement to take the abstract queue to a more concrete representation using circular buffer, that is an array r of $N+1$ cells and two counters h and t for which $h = 0$ and $t =$ -1 when r is empty, otherwise h *mod (N+1)* and t *mod (N+1)* the head and tail indexs of r which represents the head and tail of the queue.
Using the Reynolds method, the coupling invariant would be:

$$
\begin{array}{ll}
C : Q(V_t) \times V_t^* \times \mathbb{N} \times \mathbb{Z} & \rightarrow \mathbb{B} \\
C(<>, r, h, t) & = (t - h + 1 = 0) \\
C(< q, x >, r, h, t) & = (r[t \mod (N+1)] = x \wedge C(q, r, h, t)) \\
C(< x, q >, r, h, t) & = (x = r[h \mod (N+1)] \wedge C(q, r, h, t))
\end{array}
$$

For INITIALISE operations:

$$\llcorner q, r, h, t : [\text{TRUE}, q =<> \wedge C(<>, r, h, t)]\lrcorner_{(I_1)}$$

$$
\begin{array}{ll}
(I_1) \sqsubseteq & \langle \textbf{sc, c-frame...} \rangle \\
& \llcorner q : [\text{TRUE}, q =<>]\lrcorner_{(I_2)} \\
& ; \llcorner r, h, t : [q =<>, q =<> \wedge C(<>, r, h, t)]]\lrcorner_{(I_3)} \\
(I_2) \sqsubseteq & \langle \textbf{initialise} \rangle \\
& q :=<> \\
(I_3) \sqsubseteq & \langle \textbf{ass} \rangle \\
& h := 0;\ t := -1
\end{array}
$$

For ENQUEUE operations:

$$\llcorner q, r, h, t : [C(q, r, h, t), q =< q_0, x > \wedge C(q, r, h, t)]\lrcorner_{(E_1)}$$

$$
\begin{array}{ll}
(E_1) \sqsubseteq & \langle \textbf{sc, c-frame...} \rangle \\
& \textbf{con}\ Q;
\end{array}
$$

$$\llcorner q : [C(Q,r,h,t) \wedge q = Q, q =< Q, x > \wedge C(Q,r,h,t)]\lrcorner_{(E_2)}$$
$$;\llcorner r,t : [q =< Q, x > \wedge C(Q,r,h,t), q =< Q, x > \wedge C(q,r,h,t)]\lrcorner_{(E_3)}$$

$(E_2) \sqsubseteq$ $\langle$**enqueue**$\rangle$
ENQUEUE(q, x)
$(E_3) \sqsubseteq$ $\langle$**ass**$\rangle$
$t = t + 1;\ r[t \mod (N+1)] = x$

For DEQUEUE operations:

$$\llcorner x, q, r, h, t : [q =< a, b > \wedge C(q,r,h,t), x = a \wedge q = b \wedge C(q,r,h,t)]\lrcorner_{(D_1)}$$

$(D_1) \sqsubseteq$ $\langle$**sc, c-frame,w-pre...**$\rangle$
$\llcorner q, x : [q =< a, b > \wedge C(q,r,h,t), x = a \wedge q = b \wedge C(< a, b >, r, h, t)\lrcorner_{(D_2)}$
$;\llcorner r, h, x : [q = b \wedge C(< a, b >, r, h, t), x = a \wedge q = b \wedge C(q,r,h,t)]\lrcorner_{(D_3)}$

$(D_2) \sqsubseteq$ $\langle$**enqueue**$\rangle$
$x =$ DEQUEUE(q)
$(D_3) \sqsubseteq$ $\langle$**ass**$\rangle$
$x = r[h \mod (N+1)];\ h = h + 1$

6 Code

Putting the code together we have

var $q := \langle\rangle$ (1)
var $r : V_t^*$; **var** $h : \mathbb{N} := 0$; **var** $t : \mathbb{Z} := -1$ (2)
var $n : \mathbb{N} := 0$ (3)
$f :=$ FALSE (4)
ENQUEUE(q, r_t) (5)
$t := t + 1;\ r[t \mod (N+1)] := r_t$ (6)
$n := 1$ (7)
while $n \neq 0 \wedge \neg f$ **do** (8)
  var $tmp :=$ DEQUEUE(q, n) (9)

$tmp := r[h \mod (N+1);\ h := h+1$ (10)

$n := n - 1$ (11)

if $k_t(tmp) = k$ **then** (12)

$f :=$ TRUE (13)

$v := \lambda_t(tmp)$ (14)

else (15)

var $m := \Gamma_t(tmp)$ (16)

while $m \neq \emptyset$ **do** (17)

var $c :\in\ m$ (18)

ENQUEUE(q, c) (19)

$t := t+1;\ r[t \mod (N+1)] := c$ (20)

$n := n+1$ (21)

$m := m \backslash \{c\}$ (22)

od (23)

fi (24)

od (25)

Text in red are abstract queue operations. We complete the data refinement by replacing red lines with blue lines

7 MAYBE RUBBISH

include that children of tmp do not exist in q in precondition

$(d) \sqsubseteq$ ⟨**i-loc**⟩

var $m \cdot \llcorner[k_t(tmp) \neq k, q = q_0 \cdot \Gamma\ (tmp)]\lrcorner_{(d_1)}$

$(d_1) \sqsubseteq$ ⟨**s-post**⟩

$m, tmp, q : [k_t(tmp) \neq k, \Gamma_t(tmp) \subseteq q]$

$\sqsubseteq$ ⟨**seq**⟩

$\llcorner m, q, tmp : [k_t(tmp) \neq k, \Gamma_t(tmp) \subseteq m \cup q]\lrcorner_{(e)}$

$\llcorner m, tmp, q : [\Gamma_t(tmp) \subseteq m \cup q, \Gamma_t(tmp) \subseteq q]\lrcorner_{(f)}$

$(e) \sqsubseteq$ ⟨**ass**⟩

var $m := \Gamma_t(tmp)$

$(f) \sqsubseteq$ ⟨**while**⟩

while $m \neq\ \emptyset$ **do**

$\llcorner m, tmp, q : [\Gamma_t(tmp) \subseteq m \cup\ q \wedge m \neq \emptyset, \Gamma_t(tmp) \subseteq m \cup\ q]\lrcorner_{(g)}$

od

$(g) \sqsubseteq$ $\langle$**i-loc**$\rangle$

var $c \cdot \llcorner c, m, tmp, q : [\Gamma_t(tmp) \subseteq m \cup\ q \wedge m \neq \emptyset, \Gamma_t(tmp) \subseteq m \cup\ q] \lrcorner_{(h)}$

$(h) \sqsubseteq$ $\langle$**seq**$\rangle$

$c, m, tmp, q : [\Gamma_t(tmp) \subseteq m \cup\ q \wedge m \neq \emptyset, \Gamma_t(tmp) \subseteq m \cup\ q]$

We prove that the precondition to the procedure search implies the invariant

$$
\begin{aligned}
& I[{}^{1}/_{n}][{}^{\langle\rangle}/_{q}][{}^{\text{FALSE}}/_{f}][{}^{\langle\rangle}/_{q}] \\
& = \exists j \in \mathbb{N} \cdot (\forall\ x \in \Gamma^{\forall\ i<j}(r_t) \cdot K_t(x) \neq k) \\
& \wedge 0 \leq\ 1 \leq\ N \wedge \langle\rangle \subseteq \Gamma_t^0(r_t) \\
& \wedge (\forall y \in V_t(y \in \Gamma_t^*(r_t) \wedge y \notin \Gamma_t^+(y))) \\
& \wedge \forall\ p \in \Gamma_t^0(r_t) \wedge p \notin\ \langle\rangle \wedge k_t(p) \neq k)
\end{aligned}
$$

most of the invariant is made redundant by the fact that the q is substituted by an empty set and the because f is substituted by FALSE. The only value that j can be is 0.