Assignment 3 COMP2111 17s1
Bounded Buffer vs Queue

Kai Engelhardt

Revision: 1.8 of Date: 2017/05/29 10:21:03

This assignment is worth 17 marks and due before the end of week 11 + 10 + 1 days extension, that is, Thursday June 1st, 23:59:59 local time Sydney. Assignments should be done in pairs.

Summary

You're given a specification of a program that searches for an element in a tree. You'll implement that specification by deriving a breadth-first-search (*BFS*) solution that uses an abstract bounded queue. Then you'll use Reynolds' data refinement recipe to refine this code to one that uses a circular buffer. Finally, you'll translate that code to a C function.

Next we introduce some technical terms that we can then use to specify searching in a tree.

Definitions

We shall work with trees that have nodes labelled with keys in K and payloads in T, where K and T are some sets. More formally, a (K, T)-*graph* g is characterised by a 5-tuple $g = (V, \Gamma, r, \kappa, \lambda)$, where

- V is a set of *nodes*,
- $\Gamma : V \longrightarrow \mathcal{P}(V)$ is a *successor function*,
- $r \in V$ is a *root* node,
- $\kappa : V \longrightarrow K$ is a *key function* for the nodes, and
- $\lambda : V \longrightarrow T$ is a *payload function* for the nodes.

We call g a (K,T)-*tree* if it satisfies $\textsc{Tree}(g) = (\Gamma^*(r) = V \wedge \forall v \in V\,(v \notin \Gamma^+(v)) \wedge \forall v, w \in V\,(v \neq w \Rightarrow \Gamma(v) \cap \Gamma(w) = \emptyset))$, that is, every node is reachable from the root, there are no cycles, and all nodes have unique parents.

If g is a (K,T)-graph we refer to its components by subscripting with its name g, that is, $g = (V_g, \Gamma_g, r_g, \kappa_g, \lambda_g)$.

Now we're prepared to specify the search procedure.

$$\textbf{proc}\ \textsc{Search}(\textbf{value}\ t, \textbf{value}\ N, \textbf{value}\ k, \textbf{result}\ v, \textbf{result}\ f) \cdot$$
$$t, N, k, v, f : \left[\begin{array}{l} \textsc{Tree}(t) \wedge \max_{i \in \mathbb{N}} |\Gamma_t^i(r_t) \cup \Gamma_t^{i+1}(r_t)| \leq N, \\ (f \wedge \exists w \in V_{t_0}\,(\kappa_{t_0}(w) = k_0 \wedge \lambda_{t_0}(w) = v)) \vee \\ (\neg f \wedge \forall w \in V_{t_0}\,(\kappa_{t_0}(w) \neq k_0)) \end{array}\right]$$

where

- t is a (K,T)-tree,
- $N \in \mathbb{N}$ is a bound,
- $k \in K$ is the search key,
- $v \in T$ is to be set to the payload of a tree node that has the key k if such a node exists, and
- $f : \mathbb{B}$ is a flag that's set to **1** iff the search was successful.

The second conjunct of the precondition is a rather technical but sufficient condition for BFS on t never needing to store more than N nodes in the queue. The postcondition states that f indicates whether the search was successful and, if it was, v contains a correct search result.

BFS implementations typically employ a queue. This queue stores a *search frontier*, which consists of unexplored nodes at roughly the same distance from the root as the current focus of the search. To be more precise, the queue contains some nodes at the same distance, say, i, followed by nodes at distance $i+1$ that are not successors of the older nodes in the queue.

Given a size $N \in \mathbb{N}_{>0}$, an *N-bounded queue* is data structure that stores up to N values in a first-in-first-out manner. The operations on N-bounded queues that we need for BFS are the following.

initialise: initialises a queue that can hold up to N elements to the empty queue value.

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

dequeue: return the oldest item in a queue and remove it from the queue.

isempty: return whether a queue is empty.

Tasks

1. Use the refinement calculus to derive a BFS implementation of the SEARCH procedure specification that uses an abstract bounded queue q and a counter n for the length of q.

2. Use Reynolds' recipe for data refinement to transform the code derived in 1, which uses representation variables q and n, to one that uses a circular buffer, that is, an array r of $N + 1$ cells plus two pointers/counters that are used to remember where the head and tail of the queue are.

3. Translate your final code of 2 into a C function search that works with the provided test harness. It may be advisable to have separate C functions for the queue operations on the circular buffer, especially those that are used more than once. C header information should be added to `bfs.h` while code should go into `bfs.c`.

4. Describe your solutions to tasks 1–3 in a LaTeX document that your tutor enjoys reading. In more detail:
   - Display all your derivation work and the resulting code artifacts.
   - List, and discharge by proof, all proof obligations arising from the derivations and the data refinement.
   - Justify any changes made during the translation to C.

Deliverables

`bfs.h` C header file.

`bfs.c` C source.

`bfs.tex` is a LaTeX document with your names or student numbers mentioned in the `\author` command. It contains your task 4 solution.

Testing

Examples of the interaction with your C source are as follows. (Our shell prompt is `$` and user input is coloured red.)

```
$ make all pdf
$ ./bfstest 5 < ../tests/t1.txt
1: 1.110000,
[
  4: 4.440000,
  [
  ]
```

```
  3: 3.330000,
  [
    5: 5.550000,
    [
      6: 6.660000,
      [
      ]
    ]
  ]
  2: 2.220000,
  [
  ]
]
Found payload 1.110000 for key 1.
Found payload 2.220000 for key 2.
Found payload 3.330000 for key 3.
Found payload 4.440000 for key 4.
Found payload 5.550000 for key 5.
Found payload 6.660000 for key 6.
Key 7 could not be found.
1: 1.110000,
[
  4: 4.440000,
  [
  ]
  3: 3.330000,
  [
    5: 5.550000,
    [
      6: 6.660000,
      [
      ]
    ]
  ]
  2: 2.220000,
  [
  ]
]
```

Submission Instructions

When submissions are enabled, the `give` command to be run is:

```
% 2111
% give cs2111 ass3 bfs.h bfs.c bfs.tex
```

Do not submit any of the provided styles or C files. The provided subject-specific LaTeX style files and the assignment-specific files will be copied into the directory where your submission is built. It is your responsibilty to test whether your submission (both .c and .tex) works as expected on CSE machines.

```
$Log: ass3.tex,v $
Revision 1.8  2017/05/29 10:21:03  kaie
Summary: yet another weak (not week) deadline extension. where is this going?
         have i lost all sense of reasonable standards?

Revision 1.7  2017/05/17 03:31:23  kaie
Summary: third is now second (thx Liam)

Revision 1.6  2017/05/16 09:03:05  kaie
Summary: trees aren't trees unless we say that they are

Revision 1.5  2017/05/13 03:33:39  kaie
Summary: de-formalised queue ops

Revision 1.4  2017/05/13 03:22:42  kaie
Summary: deadline adjusted

Revision 1.3  2017/05/11 01:43:52  kaie
Summary: search spec argments match prototype in bbq.h

Revision 1.2  2017/05/09 12:40:23  kaie
Summary: bfs

Revision 1.1  2017/05/05 06:37:18  kaie
Initial revision
```