McMaster University
Department of Computing and Software
Musa Al-hassy

COMP SCI 3EA3 — Software Specification and Correctness

March 16, 2017

Exercise 10.0 Order Properties

1. Formalise the following *order properties* for a "problem", say $p : \mathbb{N} \to \mathbb{B}$, where YES and NO are synonymous with *true* and *false*.

   > Getting a YES answer for some potential solution means you would also get a YES answer for any candidate after it.

   and

   > If you get a NO answer (for a candidate), you would get a NO answer for any candidate before this one as well.

   and

   > The problem (p) has answers being a sequence of NO's followed by a sequence of YES's.

2. Explain what the following statement says —ie give a *colloquial English* phrasing of it— and prove or disprove it.

   $$p[M..N] \text{ antitonic} \quad \equiv \quad \exists i : M..N \bullet \quad \mathsf{all}\ p[M..i] \quad \wedge \quad \mathsf{all}\ (\neg p)[i+1..N]$$

   where $\mathsf{all} : \mathsf{Array}\,\mathbb{B} \to \mathbb{B}$ with: $\mathsf{all}\ q[a..b] \quad \equiv \quad \forall i : a..b \bullet q[i]$; and of-course $(\neg q)[i] = \neg(q[i])$.

3. Are the following equivalent for $p : \mathbb{N} \to \mathbb{B}$? Prove if possible and provide counterexamples otherwise.

   - p monotonic
   - $\forall x \bullet \quad p\,x \Rightarrow \quad p(x+1)$
   - $\forall x \bullet \neg p\,x \Rightarrow \neg p(x-1)$

   What if $p : \mathbb{Z} \to \mathbb{B}$?

4. Given $f : \mathbb{N} \to \mathbb{R}$ and $l : \mathbb{R}$, let $p(x) :\equiv f\,x \geq l$.

   Is p antitonic? What if f is monotone, then what can be said about the order properties of p?

5. Suppose we have antitonic problem $b : \mathbb{N} \to \mathbb{B}$ and

   > we are investigating a closed interfal which surely contains the last solution to b

   Then, if the middle of the interval is a *true* via b, then can we discard the first half of the interval since the *last* solution cannot possibly be there? What if the middle is *false*?

6. In Binary Search, computing the midpoint using the sum '$x + y$' in $(x + y) \div 2$ could result in overflow, is it the case that this can be avoided completely by instead using $m := x + (y - x) \div 2$? If so, why. If not, explain and suggest other improvements.

7. In Binary Search, we have been using '$\div$' as rounding-down, as is specified by its Galois definition. How must this be adjusted if the implementation language rounds division upwards? Is there a rounding-agnostic expression we would use instead? —read the textbook!

8. If $b[M..N]$ is antitonic, then is it the case that $\neg b[M] \equiv (\forall i : M..N \bullet \neg b\,i)$?

   Express this in colloquial English and it may be easier to "see" whether it is true or not; better yet, calculate using the lattice quantification theorems!

Exercise 10.1 Binary Search

The following is a list of problems solvable by Binary Search —as presented in the theorems list.
Formulate a relation $\mathcal{Z}$ —or a predicate b— for each instance.

1. Number of times a given element occurs in an $\leq$-ordered array.
2. Finding a fixpoint —$A[i] = i$— of a $<$-ordered array A.
3. Computing $\log_2 x$ for a given $x : \mathbb{N}$.
   Computing $f\ x$ where $f : \mathbb{N} \to \mathbb{R}$? What properties on f are needed, if any?
4. Finding a local minima of an array —an i with $A[i-1] > A[i] < A[i+1]$.
5. Finding out wheather a given array $A[0..N-1]$ is a rotated sorted array
   —computing $\exists k \bullet \forall i \bullet A[(i+k) \mod N] \leq A[(i+1+k) \mod N]$
6. Finding the k-th element of the sorted union of 2 sorted lists.
7. As an efficient strategy for the "guess a number" game.
   If you're thinking of a number between 1 and 4 billion, what is the worst-case number of guesses one might make using linear search and what is the ridiculously obscene improvement if one uses Binary Search?
8. Efficiently finding a typo in code —more generally, git biscet.
9. Implementing fast autocompletion in an editor.
10. Efficiently finding a solution to an antitonic problem.

Finally, Binary Search is fast since we break the search space in-half at each stage. What if we broke it into more sections each time, would it be faster?
Notice that if we break the array into N-intervals where N is the length of the given interval of our underlying array, then we essentially have Linear Search!

Exercise 10.2 — Report: Optional Assignment, 5%

Do the previous "Optional Assignment", at its original worth,

OR

Propose your own assignment to the instructor, possibly of greater worth,

OR

Write a report on "A Survey of Binary Search" in which you compare and contrast a traditional variant of Binary Search —any Google search or algorithms text presentation— with the general one presented in this class or an instance of our general algorithm.
Your report ought to

- Briefly explain how both chosen variants work.
- Why it is difficult to remember the traditional variant whereas the other is easier to remember and derive.
- Why they look so different in shape —for example, the different loop guards, the +/-1's in the body, and the indirect method of computing the midpoint.
- Debunk the myth that the traditional variant is better since it can exit early via a `return` or a `break` command. Not only are they more complicated, traditional variants are less efficient since they may terminate *later* than the general variant for arrays of more than a few elements!

Write and submit a report similar to the indications on Sheet 5.