3EA3 notes - Constructing Bounded Linear Search

Tanya Bouman

February 27, 2017

1 Review: Obtaining Invariants

If a requirement R can be written as $P \wedge B$, where P is easily truthified and B is more difficult to truthify, then the program is as follows:

$\{G\}$
truthify P;
do $\neg B \rightarrow$??? od
$\{P \wedge B\}$
$\{R\}$

This is known as "Deleting a Conjunct" in the theorems list.

2 Bounded Linear Search

2.1 Requirements for Bounded Linear Search

Bounded Linear Search yields the least index $i : 0..N-1$ for which $b\,i$ holds, and returns N if no such index exists —i.e. when $b[0..N-1]$ is constantly *false*.

A naive attempt at the specification is

$$R \quad : \quad x = \downarrow i : 0..N \bullet b\,i$$

However we are not guaranteed that b is defined outside the range $0..N-1$. Let us attempt to massage the quantification so that allowing i to be N does not cause a problem.

$$
\begin{array}{ll}
& \downarrow i : 0..N \mid b\, i \qquad\qquad \text{– troublesome since } b\, N \text{ undefined} \\
= & \quad \{ \text{ Meet is greatest lower bound } \} \\
& \uparrow i : 0..N \mid (\forall j : 0..N-1 \mid b\, j \bullet i \leq j) \\
= & \quad \{ \text{ Trading and Order Properties on } \mathbb{Z} \} \\
& \uparrow i : 0..N \mid (\forall j : 0..N-1 \mid j < i \bullet \neg b\, j) \\
= & \quad \{ \text{ typing and order properties } \} \\
& \uparrow i : 0..N \mid (\forall j : 0..i-1 \bullet \neg b\, j) \qquad \text{– } b\, N \text{ is never evaluated now}
\end{array}
$$

Hence,

$$R \quad : \quad x = \uparrow i : 0..N \mid (\forall j : 0..i-1 \bullet \neg b\, j)$$

With the specification dealt with, we now turn to obtaining an invariant and we do so by breaking up R into a few conjuncts:

$$
\begin{array}{ll}
& R \\
= & \quad \{ \text{ local characterisation of integer extrema} \\
& \quad \text{—the proviso checks are left to the reader! } \} \\
& 0 \leq x \leq N \wedge (\forall j : 0..x-1 \bullet \neg b\, j) \wedge (\exists j : 0..(x+1)-1 \bullet \neg\neg(b\, j)) \\
= & \quad \{ \text{ the } \exists \text{ says some b[0..x] is true and the } \forall \text{ says all } b[0..x-1] \text{ are false} \\
& \quad \text{hence the } \exists \text{ is really only saying } b[x] \text{ is true.} \\
& \quad \text{—the reader would do well to formally prove this! } \} \\
& 0 \leq x \leq N \wedge (\forall j : 0..x-1 \bullet \neg b\, j) \wedge b\, x
\end{array}
$$

Uh-oh, what if $x = N$, then $b\, x$ is undefined!

We handle this by extending the domain of b **however** the extension can never appear in the program code; only in formulae.

Define:
$\overline{b} : \mathbb{N} \to \mathbb{B}$
$\overline{b}$ = if $i < N$
then $b[i]$
else true
fi
Use $\overline{b}$ for clarity.

Pick x to truthify R:
$x = 0; x = N$
Two different values for different parts of R!!
Split up R into P and B, according to the heuristic "Conflict Resolution".
$P : 0 \leq x \leq y \leq N \wedge (\forall j : 0..x-1 \bullet \neg b\, i) \wedge \overline{b}\, y$
$B : x = y$
As an exercise, verify that $P \wedge B \Rightarrow R$.

So we can write the program:
$\{0 \leq N\}$
$x, y := 0, N$
; do $x \neq y \rightarrow$
  if $\neg b\, x \rightarrow x := x + 1$
  fi $b\, x \rightarrow y := x$
od
$\{P \wedge \neg\neg B\}$
$\{R\}$

3 Example Using Bounded Linear Search

Given:

$$arr[0..N-1] : \mathbb{Z}$$

$$s : \mathbb{Z}$$

write a program which finds the first index where s lives or returns the length of the array.

$x, y := 0, N$
; do $x \neq y \rightarrow$
  if $arr[i] \neq s \rightarrow x := x + 1$
  fi $arr[i] = s \rightarrow y := x$
od
This is an instance of bounded linear search.

4 Possible Quiz Question for March 6

Implement as linear search and prove correct by linear search:
$x \div y$ = largest z with $z * y \leq x$

$x \div y$ = least w with $(w+1) * y > x$

Also:

- Know $\exists$-introduction and it's relation to $\sqcup$-introduction in the theorem list
- Be able to derive simple algorithms such as summation, exponentiation, division, mod, base 2 logarithm, linear search
- Know the laws of a monoid.