Science of Computer Programming 14 (1990) 243–254

Searching by Elimination

Anne Kaldewaij Berry Schoenmakers*

Department of Mathematics and Computing Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands

1 Introduction

In this paper we derive three program schemes which are applicable to a wide class of searching problems. The first scheme, which is called Searching by Elimination, solves a rather general searching problem. The two other schemes, Symmetric Linear Search and Left-to-right Linear Search, are refinements of this general scheme. For a specific problem, these schemes may be refined—using mathematical properties of the objects involved—to a program. We shall illustrate this by some examples, ranging from elementary problems to more advanced problems. It turns out that the Symmetric Linear Search scheme yields—compared to the conventional Left-to-right Linear Search scheme—more elegant programs.

The general problem setting is as follows. We are given a finite set W and a boolean function F on W, satisfying precondition

$$Q: \quad (\exists\, w : w \in W : F.w),$$

where the dot denotes function application. We are asked to derive a program that establishes postcondition

$$R: \quad F.x$$

for variable x.

We often encounter situations in which W is a consecutive subset of the integers, i.e., W is of the form $[M..N]$, where $M \leq N$. For this specific form of W we derive two additional program schemes.

2 Derivation of the program schemes

Let us first solve the general problem.

As a first step, we introduce predicate $\mathcal{F}$ defined on *subsets* of W by

*Current address: CWI, Kruislaan 413, 1098 SJ, Amsterdam, The Netherlands. E-mail: berry@cwi.nl

$$\mathcal{F}.V \;\equiv\; (\exists\, v : v \in V : F.v).$$

Then precondition Q and postcondition R may be written as

$$Q:\;\; \mathcal{F}.W,$$
$$R:\;\; \mathcal{F}.\{x\}.$$

This formulation of Q and R leads to the introduction of a set variable V and accompanying invariant P which is a generalization of both Q and R:

$$P:\;\; \mathcal{F}.V \;\wedge\; V \subseteq W.$$

By construction, this invariant may be initialized by $V{:=}W$, and from $P \wedge |V|{=}1$ we infer that the unique element of V satisfies R. Choosing $|V|$ as a variant function, this leads to

```
  V := W
; do |V| ≠ 1 → "decrease |V| under invariance of P " od
; x :="the unique element of V" .
```

From $P \wedge |V|{\neq}1$ we conclude that $|V|{\geq}2$. Our strategy is based on the fact that at least one of each two elements of V may be removed from V without violating P. This yields the following approximation for which suitable guards A and B still have to be derived:

```
  V := W
; do |V| ≠ 1 →
      "choose a and b in V, such that a ≠ b"
      { a ∈ V ∧ b ∈ V ∧ a ≠ b ∧ 𝓕.V }
    ; if A → V := V \ {a}
       [] B → V := V \ {b}
      fi
  od
; x :="the unique element of V" .
```

To guarantee the invariance of the first conjunct of P, guard A must satisfy

$$A \;\wedge\; a \in V \;\wedge\; b \in V \;\wedge\; a \neq b \;\wedge\; \mathcal{F}.V \;\Rightarrow\; \mathcal{F}.(V \setminus \{a\}).$$

To this end, we calculate a candidate for A as follows, assuming $a \in V \wedge b \in V \wedge a \neq b$:

$$
\begin{array}{ll}
 & \mathcal{F}.V \Rightarrow \mathcal{F}.(V \setminus \{a\}) \\
= & \{\text{ definition of } \mathcal{F}\text{; split off } F.a,\ a \in V \;\} \\
 & F.a \;\vee\; \mathcal{F}.(V \setminus \{a\}) \Rightarrow \mathcal{F}.(V \setminus \{a\}) \\
= & \{\text{ predicate calculus }\} \\
 & F.a \Rightarrow \mathcal{F}.(V \setminus \{a\})
\end{array}
$$

$$
\begin{array}{ll}
= & \{ \text{ definition of } \mathcal{F}\text{; split off } F.b,\ b \in V \setminus \{a\} \ \} \\
& F.a \Rightarrow F.b \ \vee \ \mathcal{F}.(V \setminus \{a,b\}) \\
\Leftarrow & \{ \text{ predicate calculus } \} \\
& F.a \Rightarrow F.b.
\end{array}
$$

Hence, a candidate for A is $F.a \Rightarrow F.b$. By symmetry, $F.b \Rightarrow F.a$ is a candidate for B, and, obviously, the disjunction of these candidates is true. We thus obtain the following program scheme:

Searching by Elimination

$\{ (\exists\, w : w \in W : F.w) \}$
$|[$**var** V : **set of** W;
  $V := W$
 ; **do** $|V| \neq 1 \ \rightarrow$
   $|[$**var** $a, b : W$;
    "choose a and b in V, such that $a \neq b$"
   ; **if** $F.a \Rightarrow F.b \ \rightarrow V := V \setminus \{a\}$
    $[\!]$ $F.b \Rightarrow F.a \ \rightarrow V := V \setminus \{b\}$
    fi
   $]|$
  od
 ; $x :=$"the unique element of V"
$]|$
$\{ F.x \}$.

The guard $F.a \Rightarrow F.b$ is most easily remembered as "b is at least as good as a", and, of course, the other guard may be coupled to the mnemonic "a is at least as good as b".

Now we are going to refine the above scheme for $W{=}[M..N]$, with $M{\leq}N$. Using the fact that in this case W and hence V are linearly ordered, we observe that $a{\in}V \wedge b{\in}V \wedge a{\neq}b$ is most symmetrically established for a and b by the assignment $a, b{:=}min.V, max.V$ (using $|V|{\geq}2$). A little thought reveals that with this choice for a and b, $V{=}[min.V..max.V]$ is kept invariant (initially: $V{=}[M..N]$). This enables us to represent set V in a succinct way by taking $V{=}[a..b]$ as an additional invariant, as a consequence of which assignment $a, b{:=}min.V, max.V$ can be replaced by **skip**. More importantly, variable V now becomes superfluous and can therefore be removed. This results in the following program scheme:

Symmetric Linear Search

$\{ (\exists i : M \le i \le N : F.i) \}$
$|[$ **var** a, b : int;
  $a, b := M, N$
 ; **do** $a \ne b \rightarrow$
    $\{ M \le a < b \le N \}$
    if $F.a \Rightarrow F.b \rightarrow a := a + 1$
    $[\!]$ $F.b \Rightarrow F.a \rightarrow b := b - 1$
    fi
  od
 ; $x := a$
$]|$
$\{ F.x \}$.

From the above it follows that the Symmetric Linear Search scheme emerges when one chooses a and b in a symmetric way. The well-known asymmetric scheme emerges from the choice $a, b{:=}min.V, min.(V \backslash \{min.V\})$, i.e., choosing the two smallest elements of V. Taking $V{=}\{a\} \cup [b{+}1..N] \ \wedge \ M{\le}a{\le}b{\le}N$ as additional invariant in the Searching by Elimination scheme then leads to:

Left-to-right Linear Search

$\{ (\exists i : M \le i \le N : F.i) \}$
$|[$ **var** a, b : int;
  $a, b := M, M$
 ; **do** $b \ne N \rightarrow$
    $b := b + 1$
    $\{ M \le a < b \le N \}$
   ; **if** $F.a \Rightarrow F.b \rightarrow a := b$
    $[\!]$ $F.b \Rightarrow F.a \rightarrow$ **skip**
    fi
  od
 ; $x := a$
$]|$
$\{ F.x \}$.

Note that in the worst case the number of assignments for this scheme is roughly twice the number of assignments for the Symmetric Linear Search scheme.

Science of Computer Programming 14 (1990) 243–254

3 Applications

In this section we derive programs for the following four problems: maxlocation, starting pit location, the celebrity problem and the bounded linear search problem.

The starting-point for each derivation is the Symmetric Linear Search scheme. Throughout this section, F is a boolean function on $[M..N]$, and a, b, and x are elements of $[M..N]$—as in the program scheme. We use i, j, and k as dummies ranging over $[M..N]$. Furthermore, $M{=}1$ in the second and third problem, and $M{=}0$ in the last problem. Finally, we sometimes use that $a{\neq}b$, or even $a{<}b$, is implied by the precondition of the selection statement in this program scheme.

Another remark regarding the derivations is the following. In order to refine the guards from a program scheme, we just try to simplify $F.a \Rightarrow F.b$. Often, this also results in a refinement for $F.b \Rightarrow F.a$, because a and b may freely be interchanged. The refinements of the guards may be stronger than the abstract guards and therefore we have to check whether the disjunction of the guards thus found is true. In some examples, however, $F.a \Rightarrow F.b$ is awkward to work with. Since $F.a \Rightarrow F.b \;\equiv\; \neg F.a \;\vee\; F.b$, we then refine $\neg F.a$ (or $F.b$) first.

3.1 Maxlocation

The problem is to locate an occurrence of the maximum value in an array. A formal specification of *maxlocation* reads (cf. [1]):

$$
\begin{array}{l}
|[\,\textbf{con}\ M, N : \text{int}\ \ \{\, M{\le}N \,\};\\
\qquad f : \textbf{array}\ [M..N]\ \textbf{of}\ \text{int};\\
\ \ \textbf{var}\ x : \text{int};\\
\quad\ \ maxlocation\\
\quad\ \ \{\, (\forall\, i :: f.i \le f.x) \,\}\\
]|.
\end{array}
$$

To use the Symmetric Linear Search scheme, we instantiate F as follows

$$F.x \;\equiv\; (\forall\, i :: f.i \le f.x).$$

We compute:

$$
\begin{array}{cl}
 & F.a \Rightarrow F.b\\
= & \{\ \text{definition of } F\ \}\\
 & (\forall\, i :: f.i \le f.a) \Rightarrow (\forall\, i :: f.i \le f.b)\\
\Leftarrow & \{\ \text{transitivity of} \le\ \}\\
 & f.a \le f.b.
\end{array}
$$

By symmetry, $F.b \Rightarrow F.a$ follows from $f.b \le f.a$.

We thus arrive at the following solution for *maxlocation*:

$$
\begin{array}{l}
|[\,\mathbf{var}\ a, b : \mathrm{int};\\
\quad a, b := M, N\\
\ ;\mathbf{do}\ a \neq b\ \rightarrow\\
\qquad \mathbf{if}\ f.a \leq f.b\ \rightarrow a := a + 1\\
\qquad [\!]\ f.b \leq f.a\ \rightarrow b := b - 1\\
\qquad \mathbf{fi}\\
\quad \mathbf{od}\\
\ ; x := a\\
]|.
\end{array}
$$

In our opinion, this is a very elegant algorithm, and it is also new to us.

3.2 Starting pit location

Another nice result is our solution to the starting pit location problem. We do not know the origin of this problem. It can be found as an exercise in [1], and it reads as follows: there are N pits located along a circular race-track. The pits are numbered clockwise from 1 through N. At pit i, there are $p.i$ gallons of petrol available. To race from pit i to its clockwise neighbour one needs $q.i$ gallons of petrol. One is asked to determine a pit from which it is possible to race a complete lap, starting with an empty fuel tank. To guarantee the existence of such a starting pit it is given that:

(1) $(\Sigma\, i :: p.i) = (\Sigma\, i :: q.i)$.

In order to state the problem concisely, we introduce $D.i.j$ as the difference of the number of gallons provided and the number of gallons needed, when racing from pit i to j in clockwise direction. More formally:

$$D.i.j = (\Sigma\, k : k \text{ between } i \text{ and } j \text{ (including } i\text{) in clockwise order} : p.k - q.k).$$

The formal specification now reads:

$$
\begin{array}{l}
|[\,\mathbf{con}\ N : \mathrm{int}\ \ \{N{\geq}1\};\\
\qquad p, q : \mathbf{array}\ [1..N]\ \mathbf{of}\ \mathrm{int}\ \ \{(1)\};\\
\ \ \mathbf{var}\ x : \mathrm{int};\\
\quad\ \textit{starting pit location}\\
\quad\ \{(\forall\, i :: D.x.i \geq 0)\}\\
]|.
\end{array}
$$

As may be expected, we define F by

$$F.x \ \equiv\ (\forall\, i :: D.x.i \geq 0).$$

We will use the following important property of D, which may be inferred from (1):

(2) $\quad D.i.k = D.i.j + D.j.k,$

for all i, j, and k in $[1..N]$. Notice that (2) also gives $D.i.i = 0$ and $D.i.j+D.j.i = 0$, for all i and j in $[1..N]$.

In order to show $(\exists\, i :: F.i)$, we derive:

$$
\begin{array}{cl}
 & F.x \\
= & \{ \text{ definition of } F \ \} \\
 & (\forall\, i :: D.x.i \geq 0) \\
= & \{ \ (2) \ \} \\
 & (\forall\, i :: D.x.1 + D.1.i \geq 0) \\
= & \{ \ (2), \text{ hence } D.x.1 = -D.1.x \ \} \\
 & (\forall\, i :: D.1.i \geq D.1.x),
\end{array}
$$

expressing that x is a starting pit, whenever $D.1.x$ is minimal. Since $D.1.x$ is minimal for some x in $[1..N]$, we conclude that the precondition in the program scheme is satisfied.

Now, we may proceed as in the previous section:

$$
\begin{array}{cl}
 & F.a \Rightarrow F.b \\
= & \{ \text{ definition of } F \ \} \\
 & (\forall\, i :: D.a.i \geq 0) \Rightarrow (\forall\, i :: D.b.i \geq 0) \\
= & \{ \ (2) \ \} \\
 & (\forall\, i :: D.a.b + D.b.i \geq 0) \Rightarrow (\forall\, i :: D.b.i \geq 0) \\
\Leftarrow & \{ \text{ arithmetic } \} \\
 & D.a.b \leq 0.
\end{array}
$$

Similarly, we have that $F.b \Rightarrow F.a$ follows from $D.b.a \leq 0$, and, since $D.b.a = -D.a.b$, the latter condition may be rewritten as $D.a.b \geq 0$

In order to obtain a linear program, we introduce variable d with additional invariant: $d = D.a.b$. Observe that d has to be initialized to $D.1.N$, which equals $-D.N.1$. Furthermore, $D.a.b = (\Sigma\, k : a \leq k < b : p.k - q.k)$, since $a \leq b$. Our solution is:

```
|[var a, b, d : int;
   a, b, d := 1, N, q.N − p.N
  ; do a ≠ b  →
       if d ≤ 0  → a, d := a + 1, d + q.a − p.a
        [] d ≥ 0  → b, d := b − 1, d + q.(b−1) − p.(b−1)
       fi
    od
  ; x := a
]|.
```

In this case, the solution obtained from the Left-to-right Linear Search scheme is also interesting:

$$
\begin{array}{l}
|[\,\mathbf{var}\ a,b,d : \text{int};\\
\quad a,b,d := 1,1,0\\
\ ;\mathbf{do}\ b \neq N \ \rightarrow\\
\qquad b,d := b+1, d+p.b-q.b\\
\quad\ ;\mathbf{if}\ d \leq 0 \ \rightarrow a,d := b,0\\
\qquad [\!]\ d \geq 0 \ \rightarrow \mathbf{skip}\\
\qquad \mathbf{fi}\\
\quad \mathbf{od}\\
\ ; x := a\\
]|.
\end{array}
$$

Compared to the other solution this program is shorter. Due to the symmetry in the Symmetric Linear Search scheme, one may in general expect that the program text for additional variables is rougly twice as long as for the Left-to-right Linear Search scheme.

3.3 The celebrity problem

This problem seems to be part of the folklore. It was posed to us a few months ago, and recently we found a treatment in [3]. We follow the description in [3]: among N persons, a celebrity is someone who is known by everyone, but does not know anyone. This relation between these persons is represented by a boolean matrix r as follows:

$$r.i.j \ \equiv \ \text{“person } i \text{ knows person } j\text{”}.$$

For the sake of convenience we assume that such a celebrity exists among these persons.

For this problem we define F beforehand as

$$F.x \ \equiv \ (\forall\, i : x \neq i : r.i.x \wedge \neg r.x.i),$$

so as to shorten the formal specification, which now reads:

$$
\begin{array}{l}
|[\,\mathbf{con}\ N : \text{int}\ \ \{\, N{\geq}1 \,\};\\
\qquad r : \mathbf{array}\ [1..N]{\times}[1..N]\ \mathbf{of}\ \text{bool}\ \ \{\, (\exists\, i :: F.i) \,\};\\
\ \mathbf{var}\ x : \text{int};\\
\quad\ \ \textit{celebrity}\\
\quad\ \ \{\, F.x \,\}\\
]|.
\end{array}
$$

We derive:

$$
\begin{array}{ll}
 & F.a \Rightarrow F.b\\
\Leftarrow & \{\ \text{predicate calculus, ignoring } F.b\ \}
\end{array}
$$

$$
\begin{array}{ll}
 & \neg F.a \\
= & \{ \text{ definition of } F\text{; De Morgan } \} \\
 & (\exists\, i : a \neq i : \neg r.i.a \;\vee\; r.a.i) \\
\Leftarrow & \{ \; a \neq b, \text{ predicate calculus } \} \\
 & \neg r.b.a \;\vee\; r.a.b.
\end{array}
$$

Similarly, we have $\neg r.a.b \;\vee\; r.b.a \Rightarrow (F.b \Rightarrow F.a)$.

Since $r.a.b \;\vee\; \neg r.a.b \;\equiv\;$ true, we may strengthen the guards slightly, thereby destroying some of the symmetry:

```
|[var a, b : int;
   a, b := 1, N
  ; do a ≠ b  →
      if   r.a.b  → a := a + 1
       [] ¬r.a.b  → b := b − 1
      fi
    od
  ; x := a
]|.
```

To appreciate the above derivation the reader is invited to study the treatment in [3].

3.4 Bounded linear search

In [2] a nice solution is presented for the bounded linear search problem. Starting from our Symmetric Linear Search scheme, we will derive the same solution. The problem is stated as follows. A boolean function d is defined on $[0..N{-}1]$, and one is asked to assign to variable x the least number i in $[0..N{-}1]$ such that $d.i$. If no such number exists, N has to be assigned to x.

The formal specification reads:

```
|[con N : int  { N≥0 };
        d : array [0..N−1] of bool;
  var x : int;
   bounded linear search
   { (∀ i : i < x : ¬d.i) ∧ d.x }
]|,
```

where we have extended d (mentally) with $d.N \equiv$ true.

As before, we choose

$$F.x \;\equiv\; (\forall\, i : i < x : \neg d.i) \wedge d.x.$$

The precondition of the selection statement in the program scheme implies $0 \leq a < b \leq N$, hence, we may freely use $d.a$ in the guards, but $d.b$ should be avoided, since $b = N$ is not excluded. As in the previous example we consider only $\neg F.a$, for which we derive:

$$\begin{array}{cl}
 & \neg F.a \\
= & \{ \text{ definition of } F\text{; De Morgan } \} \\
 & (\exists\, i : i < a : d.i) \;\vee\; \neg d.a \\
\Leftarrow & \{ \text{ predicate calculus } \} \\
 & \neg d.a.
\end{array}$$

For the second alternative, we proceed as follows:

$$\begin{array}{cl}
 & \neg F.b \\
= & \{ \text{ definition of } F\text{; De Morgan } \} \\
 & (\exists\, i : i < b : d.i) \;\vee\; \neg d.b \\
\Leftarrow & \{ \text{ avoiding } d.b \ \} \\
 & (\exists\, i : i < b : d.i) \\
\Leftarrow & \{ \ a < b\text{, predicate calculus } \} \\
 & d.a.
\end{array}$$

We obtain:

```
|[var a, b : int;
   a, b := 0, N
  ; do a ≠ b  →
       if ¬d.a  → a := a + 1
       [] d.a  → b := b − 1
       fi
    od
  ; x := a
]|.
```

This program differs slighty from the program in [2]: replacement of $b := b - 1$ by $b := a$ removes the difference. This replacement is allowed, since $d.a$ is an invariant of the repetition.

Along the same lines the following solution for *bounded linear search* may be derived from the Left-to-right Linear Search scheme:

```
|[var a, b : int;
   a, b := 0, 0
  ; do b ≠ N  →
       if ¬d.a  → b := b + 1;  a := b
```

$$\begin{array}{l}
\qquad\quad [] \;\; d.a \;\; \rightarrow b := N \\
\qquad\quad \mathbf{fi} \\
\quad\;\; \mathbf{od} \\
\;\; ; x := a \\
]|.
\end{array}$$

4 Some related searching problems

We present a few variations on the general searching problem treated in this paper.

First of all, we remark that it is not difficult to adapt the results of Section 2 so that the requirement $(\exists\, w : w \in W : F.w)$ may be dropped. Pre- and postcondition for the Searching by Elimination scheme are weakened as follows:

$$\begin{array}{ll}
Q' : & W \neq \emptyset, \\
R' : & \mathcal{F}.W \Rightarrow F.x,
\end{array}$$

and, accordingly, the invariant should be changed into

$$P' : \quad (\mathcal{F}.W \Rightarrow \mathcal{F}.V) \wedge \emptyset \subset V \subseteq W.$$

Distinguishing the cases $\mathcal{F}.W$ and $\neg\mathcal{F}.W$ readily leads to the conclusion that the abstract programs from Section 2 are still valid. Note that the adapted schemes may be used to remove the assumption that there is a celebrity among the N persons in Section 3.3.

Another variation emerges when one replaces precondition Q in the general searching problem by the stronger

$$Q'' : \quad (\exists!\, w : w \in W : F.w).$$

In this case the guards in the program schemes are equivalent to $\neg F.a$ and $\neg F.b$, respectively. Of course one may combine this variation with the previous one to obtain a scheme that returns a solution for the equation $x : F.x$ (if any), given that there is at most one solution.

As a final variation we consider the case that W is linearly ordered and one is asked to derive a program that computes a minimal solution, i.e., we have Q as precondition and postcondition R is strengthened to

$$R''' : \quad (\forall\, w : w < x : \neg F.w) \wedge F.x.$$

In this case the previous variation is applicable, since there is at most one x that satisfies R'''.

5 Concluding remarks

The obvious solution to the general searching problem from Section 1 is, of course:

Science of Computer Programming 14 (1990) 243–254

$$
\begin{array}{l}
\quad V := W \ ; \ x := \text{“any element of } V\text{”} \\
; \ \mathbf{do} \ \neg F.x \ \rightarrow \\
\qquad V := V \setminus \{x\} \ ; \ x := \text{“any element of } V\text{”} \\
\quad \mathbf{od}.
\end{array}
$$

For the case $W = [M..N]$ this scheme may be refined by

$$x := M \ ; \ \mathbf{do} \ \neg F.x \ \rightarrow x := x + 1 \ \mathbf{od}.$$

Although such a scheme may be adequate to solve some instances of the general searching problem, it is of no use for solving the problems from Section 3.

We have shown that the “Searching by Elimination” approach solves these problems adequately. Furthermore, we have shown that the Symmetric Linear Search scheme yields very elegant algorithms. Even such an old and familiar problem as maxlocation turned out to be a nice illustration. For people programming in an assembly language, it is interesting to remark that our solution for maxlocation has been translated into an assembly level routine by Fritz Henglein [5], leading to a much more efficient routine than the conventional one.

It is our experience that the approach presented in this paper is very fruitful. An important aspect of this approach is that we first solve a more abstract problem whose solution is subsequently refined to solve a more detailed problem. In this way a systematic separation of concerns is achieved.

The attentive reader will probably have noticed that in a few steps in the derivations of Section 3 we used $\Leftarrow$ instead of $\equiv$, while the latter would also be correct. For instance, it is easily verified that the guards derived for the first two applications are in fact equivalent to the abstract guards. Furthermore, it is easily seen that for the last two applications $\neg F.a$ is equivalent to $F.a \Rightarrow F.b$ (cf. the second variation in the previous section). We did not use $\equiv$ in these cases, since they require irrelevant proof obligations.

Finally, we would like to stress that the derivations in Section 3 are fairly straightforward. The complications particular to a specific problem can be dealt with in isolation, when one uses the program schemes. We give as a final example a program that nondeterministically computes a number in the range $[M..N]$. The reader is invited to write such a program without using our schemes! We obtain, since $F.x \equiv \text{true}$, the following program:

$$
\begin{array}{l}
|[\, \mathbf{var} \ a, b : \text{int}; \\
\quad a, b := M, N \\
\ ; \mathbf{do} \ a \neq b \ \rightarrow \\
\qquad \mathbf{if} \ \text{true} \rightarrow a := a + 1 \\
\qquad [\,] \ \text{true} \rightarrow b := b - 1 \\
\qquad \mathbf{fi} \\
\quad \mathbf{od} \\
\ ; x := a \\
]|.
\end{array}
$$

Science of Computer Programming 14 (1990) 243–254

Acknowledgement

This paper is an improvement of a similar treatment in [6]. We would like to thank W.H.J. Feijen for his comments on this earlier version. The calculational derivation of the guards A and B was inspired by [4].

References

1. Dijkstra, E.W., Feijen, W.H.J.
   A Method of Programming
   Addison-Wesley (1988).
2. Dijkstra, E.W., Feijen, W.H.J.
   The Linear Search revisited
   Structured Programming **10**, 1 (1989) 5–9.
3. Manber, U.
   Using induction to design algorithms
   Communications of the ACM **31** (1988) 1300–1313.
4. Feijen, W.H.J.
   A different derivation of the Schoenmakers-Kaldewaij Linear Search
   WF110, Eindhoven University of Technology (Dec. 1988).
5. Henglein, F.
   Fast Maximum Finding Routines
   Memo, Rijksuniversiteit Utrecht (Sep. 1989).
6. Kaldewaij, A., Schoenmakers, L.A.M.
   Searching by Elimination
   In: J.L.A. v.d. Snepscheut (ed.), *Mathematics of Program Construction*, LNCS **375** (June 1989) 297–306.