SMALL PROGRAMMING EXERCISES 17

M. REM

Department of Mathematics and Computing Science, Eindhoven University of Technology,
5600 MB Eindhoven, Netherlands

The two new exercises should not be too difficult. In Exercise 41 we have to compute the maximum number of visitors that are simultaneously present in a museum, given the moments of arrival and departure of all visitors.

The other exercise concerns the computation of a minimal track assignment for one-sided channel routing. It may be viewed as an extension of the previous exercise in that each visitor is assigned a track (a natural number) in such a way that the total number of tracks used is minimal and that simultaneous visitors have distinct tracks. Both problems allow \(O(N \log N)\) solutions.

Exercise 41: Museum peak attendance

A museum is visited by \(N\) people. Person \(i (0 \leq i < N)\) arrives at moment \(X(i)\), is present during interval \((t \leq X(i) \leq t < Y(i))\), and leaves the museum at moment \(Y(i)\). Our program should determine the maximum number of visitors that are simultaneously present in the museum:

\[
\begin{align*}
[N \; \text{int}; \quad \{N \geq 1\} \\
X, Y(i : 0 \leq i < N) : \text{array of int}; \\
\{(Ai : 0 \leq i < N : X(i) < Y(i)) \land X \text{ ascending}\} \\
[r \; \text{int}; \\
S \\
\{r = (\text{MAX} i : 0 \leq i < N : (Nj : 0 \leq j \leq i : X(i) < Y(j)))\} \\
\}\]
\]

Exercise 42: Track assignment for one-sided channel routing

We have to find a statement list \(S\) such that

\[
\begin{align*}
[N \; \text{int}; \quad \{N \geq 1\} \\
X, Y(i : 0 \leq i < N) : \text{array of int}; \\
\{(Ai : 0 \leq i < N : X(i) < Y(i)) \land X \text{ ascending}\} \\
[r \; \text{int}; \\
a(i : 0 \leq i < N) : \text{array of int}; \\
S \\
\end{align*}
\]
Solution of Exercise 39 (fault coverage)

Given is an acyclic network $G$. Its $N$ vertices are numbered from 0 upward. Vertices 0 through $M - 1$ are the sources of $G$. Each non-source $i$ produces the nand, i.e. $1 - x \cdot y$, of the values produced by its (not necessarily distinct) predecessors $p0(i)$ and $p1(i)$. Source $i$ produces value $A(i)$, where $A(i: 0 \leq i < M)$ is a given input pattern. We say that pattern $A$ detects a stuck at $x$, $x \in \{0, 1\}$, for vertex $i$, $0 \leq i < N$, if for input pattern $A$ one or more sinks produce a wrong value when all vertices but $i$ function correctly and vertex $i$ produces value $x$.

Our problem is to determine which stuck-at faults input pattern $A$ detects, i.e., to solve $S$ in

$$[[M, N: int; \{1 \leq M \leq N\}] \ p0, p1(i: M \leq i < N): array of int; \{p0 and p1 represent graph G\}] \ A(i: 0 \leq i < M): array of int; \{(A(i: 0 \leq i < M): 0 \leq A(i) \leq 1)\} \ [[f(i: 0 \leq i < N): array of int; \ S \{(A: 0 \leq i < N) \ f(i) = 1) = (A detects a stuck at 0 for vertex i) \ f(i) = 0) = (A detects a stuck at 1 for vertex i)\} \]$$

When I composed this problem I expected it to allow a linear solution. In the meantime, however, I have not been able to come up with one. The solution I present here has an $O(N^2)$ time-complexity and is due to A. Bijlsma.

For $0 \leq i < N$ we let $W(i)$ denote the value produced by vertex $i$ if all vertices function correctly. The purpose of the exercise is to assign to $f(i)$ either the value $1 - W(i)$ or the value $-1$, the choice depending on whether the effect of a stuck at $1 - W(i)$ of vertex $i$ is propagated to a sink. Before addressing this choice we first look at the problem of computing the values $W(i)$. For the recording of these values we introduce array $w(i: 0 \leq i < N)$.

Maintaining

$$P: \quad 0 \leq n \leq N$$

$$\wedge (A(i: 0 \leq i < n: w(i) = W(i)))$$

$$\wedge (A: n \leq i < N: w(i) = W(i) \lor w(i) = -1)$$
as the invariant of the second repetition, we arrive at the following (recursive) solution:

```plaintext
\[ n: \text{int}; \\
\text{w}(i: 0 \leq i < N): \text{array of int}; \\
n := 0; \text{do } n \neq N \rightarrow \text{w}(n) = -1; \text{ n := n + 1 od} \\
\text{; } n := 0 \{P\} \\
\text{; do } n \neq N \rightarrow T(n); n := n + 1 \text{ od} \\
\{ P \land n = N; \text{ hence, } (\forall i: 0 \leq i < N: \text{w}(i) = W(i)) \} \\
\]
```

where, for \( 0 \leq n < N \),

\[
T(n): \{P\} \\
\text{if } w(n) = -1 \land 0 \leq n < M \rightarrow w: (n) = A(n) \\
\text{ if } w(n) = -1 \land M \leq n < N \rightarrow T(p0(n)); T(p1(n)) \\
\text{ ; w: (n) } = 1 - w(p0(n)) \ast w(p1(n)) \\
\text{ if } w(n) \neq -1 \\
\rightarrow \text{ skip} \\
\{ P_n \}
\]

Since for each \( n \) the value of \( w(n) \) changes exactly once (from \(-1\) to \( W(n) \)), the execution time of this solution is proportional to \( N \). As in Exercise 38, essentially the same solution may be coded without recursion.

For \( 0 \leq i < N \) we let \( C(i) \) denote the set of vertices for which a stuck-at affects the value produced by vertex \( i \):

\[
C(i) = \{ j \mid \text{vertex } i \text{ produces } 1 - W(i) \text{ if vertex } j \text{ stuck at } 1 - W(j) \text{ (and all vertices other than } j \text{ function correctly)} \}.
\]

We then have for each source \( i \)

\[
C(i) = \{ i \}.
\]

For each non-source \( i \) with predecessors \( p \) and \( q \) we have

\[
W(p) = 0 \land W(q) = 0 \Rightarrow C(i) = (C(p) \cap C(q)) \cup \{ i \}, \\
W(p) = 1 \land W(q) = 0 \Rightarrow C(i) = (C(q) \setminus C(p)) \cup \{ i \}, \\
W(p) = 0 \land W(q) = 1 \Rightarrow C(i) = (C(p) \setminus C(q)) \cup \{ i \}, \\
W(p) = 1 \land W(q) = 1 \Rightarrow C(i) = C(p) \cup C(q) \cup \{ i \}.
\]

We extend our program for computing the values \( W(i) \) into one that computes the sets \( C(i) \) as well. To record the latter we introduce a boolean array \( B(i, j: 0 \leq i < \)
\( N \land 0 \leq j < N \) and extend invariant \( P \) into

\[
\begin{align*}
0 \leq n \leq N \\
\land (A_i: 0 \leq i < n: w(i) = W(i) \land \{ j | 0 \leq j < N \land B(i, j) \} = C(i)) \\
\land (A_i: n \leq i < N: w(i) = W(i) \land \{ j | 0 \leq j < N \land B(i, j) \} = C(i) \\
\lor w(i) = -1).
\end{align*}
\]

This requires changing procedure \( T(n) \) into \( U(n) \):

\[
U(n): \quad \text{if } w(n) = -1 \land 0 \leq n < M \\
\quad \rightarrow w(n) = A(n) \\
\quad \text{;} \quad [j: \text{int}; j := 0 \\
\quad \quad \text{;} \quad \text{do } j \neq N \rightarrow B(n, j) = (j = n); j := j + 1 \text{ od} \\
\quad \quad ] \\
\quad \square w(n) = -1 \land M \leq n < N \\
\quad \rightarrow [p, q, j: \text{int}; p, q, j := p0(n), p1(n), 0 \\
\quad \quad ; \quad U(p); U(q) \\
\quad \quad ; \quad w(n) = 1 - w(p) \land w(q) \\
\quad \quad \text{;} \quad \text{if } w(p) = 0 \land w(q) = 0 \\
\quad \quad \quad \rightarrow \text{do } j \neq N \rightarrow B(n, j) = B(p, j) \land B(q, j); j := j + 1 \text{ od} \\
\quad \quad \quad \quad \square w(p) = 1 \land w(q) = 0 \\
\quad \quad \quad \rightarrow \text{do } j \neq N \rightarrow B(n, j) = B(q, j) \land \neg B(p, j); j := j + 1 \text{ od} \\
\quad \quad \quad \quad \square w(p) = 0 \land w(q) = 1 \\
\quad \quad \quad \rightarrow \text{do } j \neq N \rightarrow B(n, j) = B(p, j) \lor B(q, j); j := j + 1 \text{ od} \\
\quad \quad \text{fi} \\
\quad \quad ; \quad B(n, n) = \text{true} \\
\quad \quad ] \\
\quad \square w(n) \neq -1 \\
\quad \rightarrow \text{skip} \\
\quad \text{fi}
\]

We now have a solution that establishes

\[ R: \quad (A_i: 0 \leq i < N: w(i) = W(i) \land \{ j | 0 \leq j < N \land B(i, j) \} = C(i)) \]

Unfortunately, the repetitions in \( U(n) \) have caused the solution to become quadratic.

What remains is the assignment to \( f \). For each \( j, 0 \leq j < N, f(j) \) should get the value \( 1 - W(j) \) if there exists a sink \( n \) such that \( j \in C(n) \) and it should get the value \(-1\) otherwise. We record in a boolean array which vertices are sinks:

\[ (A_n: 0 \leq n < N: sk(n) = (n \text{ a sink})). \]
Array sk can be given its proper value by observing that all vertices \( i \) for which

\[
\neg(\exists n: M \leq n < N: i = p0(n) \lor i = p1(n))
\]

are sinks.

Given \( R \), the assignment to \( f \) can be accomplished by a program of the following form:

\[
\text{for each } j \text{ such that } 0 \leq j < N \text{ do}
\]

\[
\text{if } (\forall n: \text{a sink: } B(n,j)) \rightarrow f: (j) = 1 - w(j)
\]

\[
\Box (\forall n: \text{a sink: } \neg B(n,j)) \rightarrow f: (j) = -1
\]

\fi

In order to smooth the coding of the above statements, we extend arrays \( B \) and \( sk \) by

\[(\forall i: 0 \leq i < N: B(N, i)) \land sk(N).\]

Thus, our final solution becomes

\[
S: \begin{array}{l}
[n: \text{int}; \\
 w(i: 0 \leq i < N): \text{array of int}; \\
 B(i,j: 0 \leq i \leq N \land 0 \leq j < N): \text{array of bool}; \\
 n := 0; \text{do } n \neq N \rightarrow w(n) = -1; n := n + 1 \text{ od} \\
 ; n := 0; \text{do } n \neq N \rightarrow U(n); n := n + 1 \text{ od} \\
 ; \{sk(i: 0 \leq i \leq N): \text{array of bool}; \\
 n := 0 \\
 ; \text{do } n \neq N \rightarrow sk: (n) = \text{true}; B: (N, n) = \text{true}; n := n + 1 \text{ od} \\
 ; sk: (N) = \text{true}; n := M \\
 ; \text{do } n \neq N \\
 \rightarrow sk: (p0(n)) = \text{false}; sk: (p1(n)) = \text{false}; n := n + 1 \text{ od} \\
 ; \{j: \text{int}; j := 0 \\
 ; \text{do } j \neq N \\
 \rightarrow n := 0; \text{do } \neg sk(n) \lor \neg B(n, j) \rightarrow n := n + 1 \text{ od} \\
 ; \text{if } n < N \rightarrow f: (j) = 1 - w(j) \\
 \Box n = N \rightarrow f: (j) = -1 \\
 \fi \\
 ; j := j + 1 \\
 \text{od} \\
 ]
\]

]
Solution of Exercise 40 (largest square under a histogram)

We are requested to find a statement list $S$ such that

\[
\begin{align*}
\{N: \text{int}; \{N \geq 0\} \}
\{X(i): 0 \leq i < N: \text{array of int}; \\
\{(A: 0 \leq i < N: 0 \leq X(i) \leq N)\}
\end{align*}
\]

\[
\{c: \text{int}; \\
S
\{c = (\text{MAX} p, q: 0 \leq p \leq q \leq N \land C(p, q): q - p)\}
\]

where $C(p, q)$ is defined by

$C(p, q) = (A: p < i < q: X(i) \geq q - p)$.

Notice that $C(p, p)$ holds for all $p$. Application of standard techniques, such as replacing constant $N$ by a variable, yields the following invariants:

$P0$: $0 \leq n \leq N$
$\land c = (\text{MAX} p, q: 0 \leq p \leq q \leq n \land C(p, q): q - p)$,

$P1$: $d = (\text{MIN} p: 0 \leq p \leq n \land C(p, n): p)$.

Let $0 \leq n < N$. Since $C(p, n + 1) \Rightarrow C(p, n)$, we have

\[
\begin{align*}
\{(\text{MIN} p: 0 \leq p \leq n \land C(p, n): p) \\
\leq (\text{MIN} p: 0 \leq p \leq n + 1 \land C(p, n + 1): p) \\
\end{align*}
\]

Hence, $d$ is an ascending function of $n$. This allows a solution of the following form:

\[
\begin{align*}
c, n, d = 0, 0, 0 \quad \{P0 \land P1\} \\
; \text{do } n \neq N \\
\rightarrow \text{do } \neg C(d, n + 1) \rightarrow d := d + 1 \text{ od } \quad \{P1_{n+1}\} \\
; c := (n + 1 - d) \max c \quad \{P0_{n+1}\} \\
; n := n + 1 \quad \{P0 \land P1\} \\
\text{od} \quad \{P0 \land n = N\}
\end{align*}
\]

The fact that $C(n + 1, n + 1)$ holds, guarantees the inner repetition to terminate.

Next we address for $0 \leq d \leq n + 1$ the term $C(d, n + 1)$, which occurs in the above program. For $d = n + 1$ it holds. Let $d \leq n$. Then

\[
\begin{align*}
C(d, n + 1) \\
= \\
(A: d \leq i < n + 1: X(i) \geq n + 1 - d) \\
= \\
(A: d \leq i < n: X(i) \geq n + 1 - d) \land X(n) > n - d. \quad (1)
\end{align*}
\]

Considering the first conjunct of (1), it is tempting to introduce an invariant for
recording the value of

\((\text{MIN } i: d \leq i < n: X(i))\).

However, such an invariant would lead to an \(O(N^2)\) algorithm, because searches through the whole segment \((i: d \leq i < n)\) may be required to determine the new minimum when \(n\) is increased.

Rather, we rewrite (1) such that it contains the conjunct \(C(d, n)\), which we know (by P1) to hold:

\[C(d, n+1) = C(d, n) \land (\text{Ni: } d \leq i < n: X(i) = n - d) = 0 \land X(n) > n - d.\]

The middle conjunct in this formula suggests the introduction of the following invariant:

\(P2: (\text{Aj: } 0 \leq j \leq N: h(j) = (\text{Ni: } d \leq i < n: X(i) = j)).\)

The maintenance of \(P2\) when increasing \(d\) or \(n\) is obvious.

Given \(P1\) and \(P2\), we now have for \(0 \leq d \leq n + 1\)

\[C(d, n+1) = \begin{cases} \text{true} & \text{if } d = n + 1, \\ h(n - d) = 0 \land X(n) > n - d & \text{if } d \leq n. \end{cases}\]

Hence, the guard of the inner repetition may be written as

\(d \leq n \text{ cand } (h(n - d) \neq 0 \lor X(n) \leq n - d).\)

Substituting the guard above and adding the code required to establish and maintain \(P2\) yields the following solution:

\[
S: \quad \begin{array}{l}
|\begin{array}{l}
\text{i1: } [n, d: \text{int}]; \\
\text{i2: } h(j: 0 \leq j \leq N): \text{array of int}; \\
\text{i3: } c, n, d := 0, 0, 0 \\
\text{i4: } \text{do } j : \text{int}; j := 0; \text{do } j \leq N \rightarrow h: (j) = 0; j := j + 1 \text{ od} \text{ od } j \neq N \\
\text{i5: } \text{do } d \leq n \text{ cand } (h(n - d) \neq 0 \lor X(n) \leq n - d) \\
\text{i6: } h: (X(d)) = h(X(d)) - 1; d := d + 1 \\
\text{i7: } \text{od} \\
\text{i8: } c := (n + 1 - d) \text{ max } c \\
\text{i9: } h: (X(n)) = h(X(n)) + 1; n := n + 1 \\
\text{i10: } \text{od} \\
\text{end} \\
\end{array}\end{array}
\]

Since \(0 \leq d \leq n \leq N\) may be concluded from the invariant, the innermost repetition makes, over the whole computation, at most \(N\) steps. Our solution is, consequently, linear in \(N\).