We want to find the smallest number in 0..n with property p. Linear search solves the problem. But evaluating p is expensive; let us say it takes time 1, and all else is free. The fastest solution is to evaluate p on all n numbers concurrently, and then find the smallest number that has the property. Write a program without concurrency for which the sequential to concurrent transformation gives the desired computation.

§ We introduce array A: [n*bin]. We define the desired result R, condition Ii, and helper specification P as follows.

\[
R = \neg (\exists j: 0..h \cdot pj) \land (ph' \lor h'=n)
\]

\[
Ii = \forall j: 0..i \cdot Aj=pj
\]

\[
P = Ii \land \neg (\exists j: 0..h \cdot pj) \Rightarrow R
\]

Now the program is

\[
R \iff I0 \Rightarrow I'n. h:= 0. P
\]

\[
I0 \Rightarrow I'n \iff \text{for } i:= 0..n \text{ do } Ii \Rightarrow I'(i+1) \text{ od}
\]

\[
Ii \Rightarrow I'(i+1) \iff Ai:= pi
\]

\[
P \iff \text{if } h=n \text{ then ok else if } Ah \text{ then ok else } h:= h+1. P \text{ fi fi}
\]

The n iterations of the for-loop can be executed concurrently.

We can express the result of the sequential to concurrent transformation at source as follows.

\[
R \iff I0 \Rightarrow I'n. h:= 0. P
\]

\[
I0 \Rightarrow I'n \iff i:= 0. Ii \Rightarrow I'n
\]

\[
Ii \Rightarrow I'n \iff \text{if } i=n \text{ then ok else Ai:= pi } \| (i:= i+1. Ii \Rightarrow I'n) \text{ fi}
\]

\[
P \iff \text{if } h=n \text{ then ok else if } Ah \text{ then ok else } h:= h+1. P \text{ fi fi}
\]

To understand the execution, it might help to unroll the recursion a little: in the refinement of Ii \Rightarrow I'n, replace the recursive call Ii \Rightarrow I'n by what's called if i=n then ok else Ai:= pi \| (i:= i+1. Ii \Rightarrow I'n) fi. And maybe do the same once more.