A few comments on a correctness proof of a program for the "McCarthy Airline" reservation system.

by W. Barth and J. Loeckx
A program written in Pascal for the "McCarthy Airline" reservation system is described and its correctness is proved in [1, 2]. The present note contains a few comments on the problem and on the proof. A knowledge of Section 5.2 in [1] or Section 3 in [2] is taken for granted.

1. Comments on the problem

1.1. On the nature of the problem

A program for the McCarthy Airline reservation system is "special" in that it is a continuously interactive program; such a program maps an (unbounded) sequence of inputs into a sequence of outputs.

On the other hand the McCarthy Airline reservation problem is "trivial", because the output depends on three variables (st, wl and the input) which can only take a finite number of different values. The problem therefore corresponds to a finite automaton, - as is illustrated in the next section. Conversely, any finite automaton has the same "complexity" as the McCarthy Airline reservation system. A program for such a problem merely consists of a (more or less intelligent) coding of the (finite) transition table of this automaton.

1.2. Formulating the problem as a finite automaton

The McCarthy Airline reservation system is defined as a finite automaton of the Moore model (see e.g. [3]). The input symbols are pairs (rq, ps); each state corresponds
to a pair \((st, wl)\); the output produced by the state \((st, wl)\) is \(st\).

1.3. **A more complex problem**

An intrinsically more "complex" problem corresponds to an automaton which is not a finite one. An example of such an automaton is an automaton with 2 output symbols whose behaviour (see e.g. [3], p.296) is not a finite state language.
2. Comments on the proof

The correctness proof in [1, 2] is essentially an equivalence proof of two programs for the McCarthy Airline reservation system: the first program is written in Pascal, the second one in a recursive language (viz. LCF). This fact is striking when one tries to remake the proof for a program written in LUCID [4] or a program written in LISP rather than in Pascal: in both cases the proof is void (except for notational transformations).

A more "convincing" proof may be obtained by deriving the definition of stupdt and wlupdt directly from the transition table of the automaton. For instance:

\[
\text{stupdt} = \lambda sq \ st \ \text{wl}. \\
(\text{st} = 0) \rightarrow \\
(\text{wl} = 0) \rightarrow \\
(\text{el1 sq} = 0) \\
(\text{el2 sq} = 1) \\
\rightarrow 0, \\
0, \\
(\text{el2 sq} = 1) \\
\rightarrow 1, \\
2, \\
\text{etc.}
\]

Actually, the equivalence proof of this stupdt and the stupdt of [1, 2] consists in a trivial case study; the triviality of this proof is a direct result from the triviality of the McCarthy Airline reservation system i.e. from the fact it corresponds to a finite automaton.
References


