CSC-81-023

Example of a System which is Computation Universal but not Effectively Programmable

Michael Conrad*
and
Otto Rössler**

August 1981

* Departments of Computer Science and Biology
Wayne State University
Detroit, Michigan 48202

** Institute for Physical and Theoretical Chemistry
University of Tübingen
7400 Tübingen
West Germany
Abstract

The incorporation of a chaotic component in a computing system is incompatible with its being effectively programmable. The example presented shows that the concepts of programming suitable for biological systems may differ from those which have grown out of our experience with present day digital computers.
We show by construction that systems are possible which are computation universal, but not effectively programmable.

Definition. A computer will be called effectively programmable if it is possible to communicate a digital program to it using a digital computer (e.g. a time-bounded, space-bounded Turing machine).

The machine to which the program is to be communicated is assumed to be computation universal and represented by a universal Turing machine (UTM) consisting of a finite automaton (UFA), a movable tape on which symbols can be written, and a read-write head. The program is represented as an input to the tape and is to be coded by the set of Turing quadruples \( \{q_i x_j y_k q_f\} \). As usual the \( q_r \) belong to the finite state set of UFA, the \( x_j \) belong to the finite set of inputs (or tape alphabet), and the \( y_k \) runs over the set of outputs (alphabet plus tape moves). The data on which the program is to act is represented on the tape, as a sequence \( x_{r1} \ldots x_{rn} \). Whether or not we place time or space bounds on the machine to be programmed is immaterial to the argument.

The special feature of our system is the presence of a translating device which codes external inputs into tape symbols. The only way for the programmer to write the program and data into the tape squares is through the translator, whose outputs at the end of a certain interval of time \( \tau \) determine the symbol to be written on the tape. We take the translator as described by the Lorenz equation (Lorenz):

\[
\begin{align*}
\dot{x} &= 10y - 10x \\
\dot{y} &= 28x - y - xz \\
\dot{z} &= xy - 8/3z.
\end{align*}
\]

Fact. A system described by the Lorenz equation in a certain range of
parameters exhibits chaos, that is, shows aperiodic behavior of a kind sensitively dependent on the initial conditions (see Lorenz, 1963; Guckenheimer, 1980). Many other systems are now known to exhibit chaos, such as certain continuous chemical systems with at least three variables (Rössler, 1979). Though not necessary for the argument we choose the Lorenz equation since it is well known and considered to be an example which has a strange attractor, that is, an attractor with no embedded periodic trajectories that are attracting, so that the solutions stay unperiodic over arbitrarily long times. Abstract reaction systems for which the existence of a strange attractor can be proven (see Rössler, 1979) could have been chosen instead of the Lorenz equation.

Definition. Let \( \tau_f = |Q| \Delta t \), where \( |Q| \) is the size of the state set of the digital computer (e.g. UFA plus tape) and \( \Delta t \) the length of time required for each change of state.

Lemma. Any digital computer (e.g. space-bounded Turing machine) which computes the symbols placed on the tape by solving the equation for the chaotic translator will have periodic behavior with period \( \leq \tau_f \).

Proof. A space-bounded Turing machine has a finite number of states, therefore after a sufficient (perhaps very long) amount of time it must return to a previous state, or must reach an absorbing state. The period of the cycle cannot be greater than \( \tau_f \) since the cycle cannot contain more than \( |Q| \) states.

Theorem. A computer may be computation universal but not effectively programmable.

Proof. If the time interval \( \tau_c \) (see above), which we are free to choose, exceeds the interval \( \tau_f \), it will not be possible for the digital computer to compute the output of the chaotic translator. Even an approximate
computation is impossible under any reasonable definition of approximate, since the digital computer will have returned to one of its previous states, while the translator will not have.

Note that the theorem holds even if the simulating computer is not time-bounded. If it is time-bounded accuracy will have to be traded for time and the approximation will break down for smaller values of $\tau_c'$. If the inputs to the initial-condition-sensitive translator are known only up to a certain small number of digits depending on $\tau_c'$, the theorem will hold even for $\tau_c' \ll \tau_f'$, but in this case due to ignorance. But even in the absence of ignorance, the above construction implies that it is possible to use a deterministic process to communicate programs to a computer, yet not ever be able to know what programs are communicated to it.

If the system to be programmed has a compiler rather than an interpreter, we could say that communicating the program involves setting the state of the finite automaton, UFA. If the compiler rather than the interpreter included the chaotic component, the program typed in as the input would set the state determinately, but it would be impossible to prescribe which state is set and therefore to know what program is typed in. As in the case of the chaotic translator, UTM would be deterministically programmable, but it would be impossible to specify in advance or compute what program is communicated to it as a result of selecting the inputs.

The assumption that either the translator or the process in the compiler obeys the Lorenz equation is an idealization. If one believes the natural system is in reality a finite state system, it will compute the Lorenz equation or any equation with chaotic solutions with only a certain
degree of accuracy. Under this assumption the theorem would hold only to
the extent that the digital computer computes chaos less accurately than
the natural system it simulates, leading to a breakdown of its approxima-
tion at an earlier time. If the natural system does not obey the Lorenz
equation precisely due to noise, the situation will be worse for program-
mability. Probabilities will enter and it will not even be possible to
calculate the probability distributions that are generated.

There are good phenomenological reasons (such as the phenomenon of
turbulence) for believing that chemical and other natural systems can
exhibit deterministic chaos at least to a very good approximation. An
interesting point is that instead of assuming that the chaotic natural
system computes Eq. (1), we could take this system as standing in place
of Eq. (1). For our result to hold it is only necessary for the stand-in
system to exhibit chaos. In fact we could never hope to demonstrate an
explicit equation for such a stand-in system since no digital computer
could ever provide a justifying computation.

The purpose of our construction is to show, by an almost trivial
example, that the concept of programmability is subtle. Since $\tau_c$
can have an infinite number of values which are larger than $\tau_f$, it would
never be possible to write a finite manual for a machine which incorporates
a chaotic process. The digital computers from which our intuitions about
programming are built are in this respect unusual and remarkable systems.
Systems exist in which the nature of the relationship between input and
rule executed is very different from that to which we have become accus-
tomed on the basis of our experience with these unusual systems. It is
not unreasonable to suppose that many, if not most, systems which occur
naturally are not effectively programmable. In general, biological systems
appear to fulfill conditions which make them not effectively programmable. This is not only because of the ubiquity of chaos, but also because the folding of proteins makes each new gene an emergent primitive whose function cannot be ascertained from its structure without consulting the laws of physics (Conrad, 1974, 1979). One caveat is that caution is necessary in carrying programming intuitions gained from digital computers over to biological systems.

A second caveat derives from the fact that the construction provides a concrete example in which the issue of continuity versus discreteness bears significantly on the computing power of natural systems. According to Smale's symbolic dynamics interpretation (Smale, 1967), it should be possible to view chaotic systems as calculators which compute the digits of a different irrational number for each different initial condition (aside from a subset of periodic numbers of measure zero). The loss of effective programmability in this example can be interpreted as simply due to the fact that no finite system can compute nonperiodic numbers over an arbitrary number of digits. The only plausible candidate for physical reality which can do this is a chaotic system with continuous state variables. The example thus shows that the reality or nonreality of continuity in nature determines whether it is possible to use digital computers to simulate significant information processing tasks which might be executed by biological systems, even taking simulation in its weakest acceptable sense (cf. Conrad and Rosenthal, 1980).

It is interesting that the chaotic translator protects the privacy of the rule executed by the computer from all outside observers, including the programmer himself. It therefore protects the programmed system from being simulated or predicted by any other computer, no matter how fast.
Acknowledgment

M.C. acknowledges support from the Division of Information Systems, Office of Naval Research (Contract N00014-80-C-0365).

References


