Int. J. General Systems
1978, Vol. 4, pp. 201–209.

INPUT/OUTPUT CODINGS AND TRANSITION FUNCTIONS IN EFFECTIVE SYSTEMS

BRUNO BUCHBERGER

Universität Linz, Institut für Mathematik, Linz/Auhof, Austria

and

BERNHARD ROIDER

Universität Innsbruck, Technische Fakultät, Institut für Mathematik I, Innsbruck, Austria

(*Received December 7, 1976; in final form April 1, 1977*)

We study some aspects of what might be called a general theory of effective systems. First, we define the notion of a universal effective system and derive some theorems which give the connection of this notion with related concepts in recursive function theory. Then we characterize the class of transition functions and the sets of final states of universal effective systems. Finally, we exactly determine the role of input/output codings in effective systems and characterize the class of possible input and output codings in universal effective systems. We interpret our results in terms of computer science.

INDEX TERMS Effective systems, transition functions, input/output codings, universality, Gödel numberings, automata.

It is our purpose in this paper to contribute to what might be called a general theory of effective systems. This concept is couched in terms of recursive function theory, for which we take Rogers's book[1] as our basic reference:

DEFINITION f and g determine a (deterministic) *effective system* (or an (effective) automaton): $\leftrightarrow f$ and g are recursive functions.

We shall sometimes say: (f, g) is an automaton instead of f and g determine an automaton. We think of f as the *transition function* of the automaton, which assigns the next state $f(\xi)$ to every state (= natural number) ξ. States ξ for which $g(\xi)=0$ are called *final* or *terminal.*

The dynamics of an effective system are described by the function $[f, g]$ which is recursively defined by

$$[f,g](\xi)=\begin{cases}\xi & \text{if} \quad g(\xi)=0.\\ [f,g]f(\xi) & \text{if} \quad g(\xi)\neq 0.\end{cases}$$

(For $[f, g](\xi)=\xi'$ read: ξ' is the *final state eventually reached* by successive applications of the transition function f when starting from state ξ; $[f, g]$ is called the *global transition function* determined by f and g).

In this paper we consider deterministic effective systems only. It is clear, however, how to define a non-deterministic effective system. We would have to replace the transition function with a decidable transition relation.

We feel that much of computer science may be developed in a unified and pedagogically satisfactory manner on the basis of the concept of an effective system. Let us summarize the reasons why in our opinion the concept of an effective system is neither to narrow nor too wide for this purpose.

THE CONCEPT OF AN EFFECTIVE SYSTEM IS NOT TOO NARROW:

1) The systems most commonly studied in computer science are effective systems in our sense (for example, Turing machines and other computability formalisms, abstract automata that operationally define the semantics of programming languages, real computers, all types of automata in formal language theory, cellular spaces are deterministic effective systems; phrase structure grammars, deductive systems of formal logic, lambda-calculi, L-systems are non-deterministic effective systems).

2) Systems encountered in computer science have at most a denumerable set of states. Thus, after a suitable coding we may assume the set of states to be a subset of the natural numbers. Also, without loss of generality we may require the set of states to be all natural numbers, as numbers ξ not in the range of the

coding may be defined to be final states by putting $g(\xi)=0$.

3) In the systems used in computer science, what happens in a single step must be physically executable, at least in principle. Thus, by Church's thesis, the transition function (in the deterministic case) will be recursive.

4) In the systems used in computer science, given a state ξ, it must be decidable whether ξ is final, for an automaton is useless if the user does not know when a computation is finished. Thus, by Church's thesis, the set of final states will be decidable. In other words, the set of final states is $\{\xi|g(\xi)=0\}$ for some recursive function g. In addition, dropping the requirement that the set of final states be decidable would have an unreasonable consequence in a very specific sense. Namely, it can be shown that if f is recursive and g is such that $\{\xi|g(\xi)=0\}$ is not decidable then $[f, g]$ is not even partial recursive. But by Church's thesis the global transition function of an automaton must be partial recursive.

5) Our concept of an effective system is a specialization of the concept of an autonomous discrete time system. The additional requirement of recursiveness imposed on the transition function and the set of final states seems to be the weakest assumption that permits one to derive the basic insights into the systems of computer science; these cannot be established by set theoretical and algebraic means alone. For instance, what we shall prove about the role of input and output functions in effective systems could not be derived in the setting of general system theory.

THE CONCEPT OF AN EFFECTIVE SYSTEM IS NOT TOO WIDE:

1) Though the transition functions of the concrete effective systems of computer science normally are extremely simple functions, whereas recursive functions may be arbitrarily complex, we do not see any natural way to define in precise terms what is meant by a simple transition function. In fact, no one has so far proposed a thesis as to what is possible in one computational step in an automaton. Such a thesis could play a role similar to that of Church's thesis, which essentially characterizes what is possible by iteration of computational steps.

2) In any case a useful concept of effective system should be wide enough to comprise the interpreting automata of high level programming languages with arbitrarily complex basic functions. For this purpose it is necessary to keep the class of transition functions as wide as we did.

3) As long as the set of final states is required to be decidable, computing partial recursive functions is a non-trivial task for effective systems with even very powerful transition functions, in the sense that the number of computational steps cannot be recursively bounded in general. For otherwise the halting problem of every automaton would be decidable.

4) We emphasize that no single computability formalism (e.g. Turing's or Markov's formalism) can be a suitable framework for a general theory of the automata encountered in computer science. For, given a computability formalism, by a diagonalization procedure we can construct a new automaton that is not enumerated in the formalism. Thus, a wide enough concept of an effective system must characterize a class of objects that is recursively non-enumerable, as is the class of pairs (f, g) satisfying our definition.

In recursive function theory, though the concept of an effective system has rarely been formulated explicitly (to the best of our knowledge, Shepherdson[10] was the only writer in recursive function theory who made the concept of an effective automaton explicit), some very important aspects of a general theory of effective systems have already been studied yielding famous and fundamental results. Thus, the undecidability and incompleteness results of formal logic (see, for instance, Smullyan[9]) settle the aspect of interpreting the "states" of effective systems as statements of a descriptive language (as opposed to imperative languages like programming languages). Further, the aspect of (time) complexity of effective systems is developed in Blum's complexity theory, see Blum[5]. (In an earlier paper[3] we have shown in what sense the concept of an effective system is implicit in Blum's theory). Finally, Shepherdson[10] and Cleave[11] treat the problem of representing arbitrary recursively enumerable degrees of unsolvability by the halting, word, and confluence problems of effective systems.

In computer science, several variants of the concept of an effective system, in different contexts and terminology, have appeared independently at various places in the literature[2,3,4]. The aspects considered in these papers include the interpretation of programming languages by effective systems, simulation, programmability, criteria for the universality of effective systems, and the role of input and output in effective systems. The latter two aspects are the subject matter of the present paper, too.

In Section 1 we derive a necessary and sufficient criterion that characterizes the transition functions of universal effective systems and, in passing, show how the notion of a universal effective system compares with related notions in recursive function theory. In Section 2 we establish some theorems which clarify the influence input/output conventions have on the computational power of effective systems. We feel that these latter investigations might be of particular interest for system theory.

NOTATION, ABBREVIATIONS AND AUXILIARY DEFINITIONS

N	set of natural numbers including 0
p.r.	partial recursive
r.e.	recursively enumerable
$\bar{A}$	complement of the set A
$f(x)\downarrow$	$f(x)$ is defined
$f(x)\uparrow$	is not defined
$\mathrm{dom}(f)$	domain of function f
$rg(f)$	range of function f
$f \circ g$	composition of functions f and g
P_n	set of all n-ary partial recursive functions
R_n	set of all n-ary (total) recursive functions

If f, g are functions then we often write $fg(x)$ for $f(g(x))$. In addition, $f^{(0)}(x) := x$, and $f^{(t+1)}(x) := ff^{(t)}(x)$. Let f be a function, $A \subseteq N$. We define $f(A) := \{y | (\exists x \in A)(f(x) = y)\}$. An analogous notation will be used for n-ary functions. By a pairing function (an onto pairing function) we mean a 1–1 total recursive function from N^2 into N with decidable range (with range N). We use the notation $\langle x, y\rangle$, $z_{\cdot 1}, z_{\cdot 2}$ to denote the application of some fixed onto pairing function to $x, y \in N$, and of its first and second projection to $z \in N$, respectively. Thus we have

$$\langle z_{\cdot 1}, z_{\cdot 2}\rangle = z, \langle x, y\rangle_{\cdot 1} = x, \langle x, y\rangle_{\cdot 2} = y.$$

1. TRANSITION FUNCTIONS OF UNIVERSAL EFFECTIVE SYSTEMS

Intuitively, a (deterministic) effective system is called universal if it can compute every computable function. To make this concept precise we have to specify how an effective system can be used to compute a function. An analysis of computation in real computers and abstract computability formalisms (e.g. Turing machines) shows that computing the value $h(x)$ for a given argument x on an effective system (h being a partial function) proceeds in three stages.

Firstly, depending on x, an initial state of the effective system is determined. Secondly, starting from the initial state, the automaton repeatedly applies its transition function until, perhaps, it arrives at a final state. If and when this happens, then, thirdly, depending on the final state, the output, which should be $h(x)$ provided the initial state has been determined adequately, is read off.

It is clear that the input and output conventions in stages one and three are, in general, simple compared with the computational power of the automaton. Moreover, stage two is the only one that may be responsible for $h(x)$ being undefined, i.e. the input and output conventions must be total functions. Again, we may assume the domain and range of h to be a subset of N. Thus, by Church's thesis, it is reasonable to require that the input and output conventions are recursive functions. These very general assumptions will suffice for our purposes. We are thus led to the following definitions.

1.1. DEFINITION Let f and g determine an automaton and let $h \in P_n$.
h is *computable* on the automaton (f, g): $\leftrightarrow$

(C) there exists an "*input coding*" $\gamma \in R_n$ and an "*output coding*" $\rho \in R_1$ such that $h = \rho \circ [f, g] \circ \gamma$.

1.2. DEFINITION Ψ, κ determine a universal automaton (in short: "(Ψ, κ) is a universal automaton"): $\leftrightarrow$
(U1) (Ψ, κ) is an automaton
(U2) all p.r. functions are computable on (Ψ, κ).

It is the main goal of this section to derive necessary and sufficient conditions for the functions Ψ and κ that can occur in universal effective systems (Theorems 1.5 and 1.6). By way of preparation we first indicate a few connections between our concept of a universal automaton and well-known concepts of universality in recursive function theory (Theorems 1.3 and 1.4), which have some intuitively interesting interpretations.

1.3. THEOREM *Let (Ψ, κ) be an automaton. Then the following statements are equivalent:*

1) *All p.r. functions are computable on (Ψ, κ) (i.e. (Ψ, κ) is universal).*

2) *All unary p.r. functions are computable on* (Ψ, κ).
3) *There exists an "input coding"* $\gamma \in R_2$ *and an "output coding"* $\rho \in R_1$ *such that* $\rho \circ [\Psi, \kappa] \circ \gamma$ *is a semi-effective numbering of* P_1.
4) *There exists an "input coding"* $\gamma \in R_2$ *and an "output coding"* $\rho \in R_1$ *such that* $\rho \circ [\Psi, \kappa] \circ \gamma$ *is a Gödel numbering for* P_1.

These equivalences are straightforward consequences of the definitions. The proofs are left to the reader. As for the implication (1)→(4), recall that a Gödel numbering is itself a binary p.r. function. (For the definition of semi-effective numbering and Gödel numbering of P_1 see Rogers[1], Exercise 2–11).

Remark In computer science the notion of a Gödel numbering is a generally accepted precise substitute for the intuitive concept of a universal programming language as far as functional (as opposed to operational) semantics is concerned (see, for instance, Blum[5]). Statement (4) is thus a precise formulation of the assertion that (Ψ, κ) is capable of interpreting some universal programming language. The function γ when applied to p and x describes all preparatory work necessary for the execution of a given program p on given data x, including the compilation process. (Ψ, κ), then, may be conceived as operationally defining the semantics of the language. Thus, statement (4) connects functional and operational semantics.

One might argue that the main computational work could be absorbed into the input and output codings if we admit arbitrary recursive ρ and γ in statements (3) and (4). This is not so, since, as long as the input and output codings are required to be (total!) recursive, the number of computational steps cannot be recursively bounded in a universal automaton, whose halting problem is undecidable.

The equivalence of (3) and (4), though easy to prove, has an interesting intuitive interpretation, which has escaped attention so far, namely it has bearings on the relevance (or irrelevance) for programming practice of the distinction traditionally made between fully effective (or Gödel) numberings and semi-effective numberings of P_1. So far, it has been held by most authors that only Gödel numberings of P_1 are useful models for programming systems, since the property of being a semi-effective numbering of P_1 in the sense that all partial recursive functions have "programs" in the numbering does not guarantee the existence of compilers from all the other possible (semi-effective) programming systems to the system considered. Thus Ershov[7], e.g., argues that the practical difficulties encountered in constructing compilers for real programming languages could be of a "principal character", which could be due to the fact that there are semi-effective numberings of P_1 that are not Gödel numberings (e.g. Friedberg numberings[8]) and, correspondingly, universal automata that are not "effectively" universal. The equivalence between (3) and (4) partly explains why semi-effective non-Gödel numberings of P_1 do *not* occur in practice: in practice a semi-effective numbering Ψ of P_1 (i.e. a programming language) is implemented on an automaton (Ψ, κ), i.e. the programmer uses a decomposition $\Psi = \rho \circ [\Psi, \kappa] \circ \gamma$. In view of (4), with adequate input and output conventions this same automaton is capable of interpreting a fully effective numbering, i.e. a universal programming language, as well.

1.4. THEOREM *Let* Ψ^* *be a unary p.r. function. Then the following statements are equivalent:*

1) Ψ^* *is a universal function.*
2) *There exist* $\rho, \Psi, \kappa \in R_1$ *and* $\gamma \in R_2$, *such that* $\rho \circ [\Psi, \kappa] \circ \gamma$ *is a Gödel numbering of* P_1 *and* $\Psi^* = \rho \circ [\Psi, \kappa]$.

Proof That (2) implies (1) is immediate from the definitions (for the notion of universal function see Rogers[1], p. 54). For the converse, let us recall that for any two universal functions Ψ_1^* and Ψ_2^* there exists a recursive bijective function f such that $\Psi_1^* = \Psi_2^* \circ f$ (see Cleave[6] and Ershov[7]). Now let $\Psi^* \in P_1$ be universal. Take $\rho', \Psi', \kappa' \in R_1$, $\gamma' \in R_2$ such that $\rho' \circ [\Psi', \kappa'] \circ \gamma'$ is a Gödel numbering and define $\Psi^{*\prime} := \rho' \circ [\Psi', \kappa']$. $\Psi^{*\prime}$ is a universal function by the first part of the theorem. Hence, there exists a recursive bijective function f such that $\Psi^{*\prime} = \Psi^* \circ f$.

Define
$$\gamma := f \circ \gamma'$$
$$\Psi := f \circ \Psi' \circ f^{-1}$$
$$\kappa := \kappa' \circ f^{-1}$$
$$\rho := \rho' \circ f^{-1}.$$

Then $\Psi^* = \rho \circ [\Psi, \kappa]$ and $\Psi^* \circ \gamma = \Psi^{*\prime} \circ f^{-1} \circ f \circ \gamma' = \Psi^{*\prime} \circ \gamma'$ is a Gödel numbering, as was to be shown.

Remark The equivalence shown in Theorem 1.4. is not clear from the outset since we shall see in Theorem 2.1 that it is by no means true that *all* partial recursive functions h can be represented in the form $h=r\circ[f,g]$ $(r,f,g\in R_1)$. By Theorem 1.4 a universal function Ψ^* has an easy intuitive interpretation:

There is a universal automaton that will yield output $\Psi^*(\xi)$ when started in state ξ; in other words, Ψ^* tells us how the result depends, not on program or input, but on the initial *state* of some universal automaton.

To formulate the main result of this section we need some terminology. Let (Ψ,κ) be an automaton. We define:

$X_t:=\{\xi|[\Psi,\kappa](\xi)\downarrow\}$

(set of states that eventually lead to a *terminal* state)

$$X_c:=\{\xi|\xi\notin X_t \text{ and } (\exists t_1,t_2)(t_1\neq t_2 \text{ and } \Psi^{(t_1)}(\xi)=\Psi^{(t_2)}(\xi))\}$$

(set of states that eventually run into a *cycle*)

$$X_d:=\{\xi|\text{ for no } t_1\neq t_2 \text{ do we have } \Psi^{(t_1)}(\xi)=\Psi^{(t_2)}(\xi)\}$$

(set of states that initialize *divergent* computation paths if terminal states are neglected)

$\xi\sim\eta:\leftrightarrow(\exists t_1,t_2)(\Psi^{(t_1)}(\xi)=\Psi^{(t_2)}(\eta))$ (i.e. ξ and η have a common successor state). Note that the complement of $X_t\cup X_c$ is contained in X_d.

We now show that universality of automata can be characterized by the shape of their transition graphs, viz. what is essential for universality is the presence in the graph of infinitely many divergent components that can be effectively obtained.

1.5. THEOREM *Let $\Psi\in R_1$. Then $\rho\circ[\Psi,\kappa]\circ\gamma$ is a Gödel numbering for some $\rho,\kappa\in R_1,\gamma\in R_2$ iff there is an $f\in R_1$ such that $f(N)\subseteq X_d$ and $(\forall i,j)(i\neq j\rightarrow$*not *$f(i)\sim f(j))$ (i.e. Ψ is the transition function of a universal automaton iff there is a recursive* "selection function" *f for the non-cyclic components in the stategraph).*

Proof

Sufficiency

Idea Let κ be such that (Ψ,κ) is a universal automaton. Use the recursion theorem (Rogers[1], §11.2) and the fact that the set X_c and the relation $\sim$ are r.e. to effectively obtain a new $\xi=f(n)$ provided the values $f(i)$ have been determined for $0\leq i<n$.

Details Let $\Psi=\rho\circ[\Psi,\kappa]\circ\gamma$ be a Gödel numbering of P_1 and write $\Psi^*=\rho\circ[\Psi,\kappa]$. Assume $f(0),\ldots,f(n-1)$ have been determined (for $n=0$ assume nothing). Let $g(x,y)=0$ provided $\gamma(x,y)\in X_c$ or $\gamma(x,y)\sim f(i)$ for some $i<n$; otherwise let $g(x,y)$ be undefined. Since Ψ is a Gödel numbering, there is a $\beta\in R_1$ such that $g(x,y)=\Psi(\beta(x),y)$ for all x,y. Choose a p such that $\Psi(\beta(p),y)=\Psi(p,y)$. Let $\xi=\gamma(p,0)$. We cannot have $\xi\in X_t$ because otherwise $g(p,0)=\Psi(p,0)$ is defined, so $\xi\in X_c$ or $\xi\sim f(i)$ for some i, so $\Psi^*(\xi)=\Psi(p,0)$ is undefined. We cannot have $\xi\in X_c$ or $\xi\sim f(i)$ for some $i<n$ either, because otherwise $\Psi^*(\xi)=g(p,0)$ would be defined. Let $f(n)=\xi$. As every step in the construction of $f(n)$ (especially the choice of β and p) is recursive, f will be a recursive function, and we have seen that it will have the required properties.

Necessity

Idea We use that part of the transition graph of Ψ that comprises the successor states of all the $f(i)$ to compute a fixed Gödel numbering Ψ_0.

Details Let f be a selection function. Since not $f(i)\sim f(j)$ for $i\neq j$, f is injective. Define $\bar{f}(0):=f(0)$ and $\bar{f}(n+1):=f(\bar{m})$, where $\bar{m}:=(\min m)(f(m)>f(n))$. Then $\bar{f}\in R_1$, $\bar{f}$ is injective and strictly increasing, and $\bar{f}(N)\subseteq f(N)$.

Using $\bar{f}$ we now construct a pairing function τ in the following way:

$\tau(n,0):=\bar{f}(n)$;

$\tau(n,t+1):=\Psi^{(s)}\bar{f}(n)$, where $s=(\min s')(\Psi^{(s')}\bar{f}(n)>\tau(n,t))$

(To decide "$\xi\in rg(\tau)$" compute

$\tau(0,0),\ldots,\tau(\xi,0)$

....

....

$\tau(0,\xi),\ldots,\tau(\xi,\xi)$;

if ξ does not occur among these values then $\xi\notin rg(\tau)$.)

Now $rg(\tau)$ will serve as a "computation region" for computing the fixed Gödel numbering Ψ_0.

For this purpose first choose a complexity measure[5] for Ψ_0, i.e. a function $\Phi_0\in P_2$ such that

1) $(\forall p,x)(\Phi_0(p,x)\downarrow\leftrightarrow\Psi_0(p,x)\downarrow)$,

2) the ternary predicate $\Phi_0(p,x)=t$ is decidable.

Now define

$$\gamma(p,x) := \tau(\langle p,x\rangle, 0)$$

$$\kappa(\xi) := \begin{cases} 0 & \text{if } \xi \in rg(\tau) \text{ and } \Phi_0((\tau_1(\xi))._1, (\tau_1(\xi))._2) \\ & \qquad = \tau_2(\xi) \\ 1 & \text{otherwise} \end{cases}$$

(here, τ_1 and τ_2 are projections belonging to τ)

$$\rho(\xi) := \begin{cases} 0 & \text{if } \kappa(\xi) = 1 \\ \Psi_0((\tau_1(\xi))._1, (\tau_1(\xi))._2) & \text{otherwise.} \end{cases}$$

Then $\rho, \kappa \in R_1$, $\gamma \in R_2$ and it is easily checked that $\Psi_0 = \rho \circ [\Psi, \kappa] \circ \gamma$.

To conclude this section, we characterize the possible termination criteria κ of universal automata.

1.6. THEOREM *Let $\kappa \in R_1$. There exist $\rho, \Psi \in R_1$, $\gamma \in R_2$ such that $\rho \circ [\Psi, \kappa] \circ \gamma$ is a Gödel numbering iff $\{\xi | \kappa(\xi) = 0\}$ and $\{\xi | \kappa(\xi) \neq 0\}$ are infinite.*

Proof Define $E_\kappa := \{\xi | \kappa(\xi) = 0\}$.

Sufficiency Of course, $(\forall r)\ (\exists p, x)\ (\rho[\Psi, \kappa]\gamma(p, x)) = r)$, i.e. $(\forall r)\ (\exists \xi)\ (\rho(\xi) = r$ and $\kappa(\xi) = 0)$, hence E_κ is infinite. On the other hand, by the proof of Theorem 1.5 there is a ξ such that

$$(\forall t)(\kappa\Psi^{(t)}(\xi) \neq 0) \text{ and } (\forall t_1, t_2)$$
$$(t_1 \neq t_2 \rightarrow \Psi^{(t_1)}(\xi) \neq \Psi^{(t_2)}(\xi)),$$

hence $\bar{E}_\kappa$ is infinite.

Necessity Assume E_κ and $\bar{E}_\kappa$ infinite.

E_κ and $\bar{E}_\kappa$ are decidable. By routine methods we construct a recursive surjective pairing function $\sigma_3 : N^3 \rightarrow \bar{E}_\kappa$ with projections $\sigma_{31}, \sigma_{32}, \sigma_{33}$.

Further, we construct $\delta, \delta' \in R_1$ such that δ maps N bijectively onto E_κ and $\delta'\delta(r) = r$. Let $\Psi \in P_2$ be some Gödel numbering of P_1 and Φ be a complexity measure[5] in the sense of Blum.

We now define

$$\gamma(p, x) := \sigma_3(p, x, 0) \qquad (\in \bar{E}_\kappa)$$

$$\Psi(\xi) := \begin{cases} 0 & \text{if } \xi \in E_\kappa \\ \sigma_3(\sigma_{31}(\xi), \sigma_{32}(\xi), \sigma_{33}(\xi) + 1) & \\ \text{if } \xi \in \bar{E}_\kappa \text{ and } \Phi(\sigma_{31}(\xi), \sigma_{32}(\xi)) \neq \sigma_{33}(\xi) & \\ \delta\Psi(\sigma_{31}(\xi), \sigma_{32}(\xi)) & \text{otherwise} \end{cases}$$

$$\rho(\xi) := \delta'(\xi).$$

Of course, $\gamma \in R_2$, ρ, $\Psi \in R_1$, and $\Psi = \rho \circ [\Psi, \kappa] \circ \gamma$ by the construction of γ, Ψ, ρ.

2. INPUT/OUTPUT CODINGS

Our goal in this section is twofold. First, we want to analyze the influence input/output codings have on the class of functions computable by effective systems. We shall see that even if we consider all possible effective automata (not only some r.e. class of automata like Turing machines etc.) we cannot dispense with the additional computational power contained in input and output conventions. More specifically, we shall prove (Theorem 2.1) that there are computable functions which cannot be computed by an automaton without input coding nor by an automaton without output coding (though, of course, every computable function can be computed by an automaton with input and output coding[3]). This gives a theoretical explanation why input and output play an important role in computing practice. Not only are input and output a matter of convenience, they are indispensable for computation. This has not always been clearly seen in the literature (for instance, in Wegner[2], p. 130, it is taken for granted that the power of an output coding may always be packed into the last step of a computation).

Secondly, in this section we characterize the functions that can occur as input/output codings of universal automata (Theorems 2.2 and 2.3). Our result on output codings will show that a universal automaton must have infinitely many states that are not actually needed to encode all possible output values. This may be interpreted as a theoretical reason for indispensability of an infinite "scratch memory" in universal automata. Unfortunately, we cannot offer a similar intuitive interpretation for our result on input codings.

2.1. THEOREM *There is a p.r. function h that is neither representable in the form $h = r \circ \lfloor f, g \rfloor$ nor in the form $h = \lfloor f, g \rfloor \circ c$ with $r, f, g, c \in R_1$.*

Proof Let $S \subseteq N$ be a simple set (see Rogers[1], p. 105).

Define

$$h(x) := \begin{cases} x & \text{if } x \in S \\ \text{undefined otherwise.} \end{cases}$$

Then $h \in P_1$ with $dom(h) = rg(h) = S$. Assume first that $h = r \circ [f, g]$ with $r, f, g \in R_1$. It is not possible that for all states ξ of the automaton (f, g) the computation starting with ξ either stops or runs into a cycle, i.e. $[f, g](\xi)\downarrow$ or $(\exists t_1, t_2)(t_1 \neq t_2$ and $f^{(t_1)}(\xi) = f^{(t_2)}(\xi))$.

For, otherwise the set $dom([f, g])$, which is equal to S, would be decidable by the following procedure: given ξ, to decide whether $\xi \in dom([f, g])$, compute $f^{(t)}(\xi)$ until either a final state is reached or a cycle is detected. Hence, there will exist a ξ, such that $R: = \{f^{(t)}(\xi) | t \in N\}$ is an infinite r.e. set and all the $f^{(t)}(\xi)$ are non-terminal. Of course, R is contained in the complement of $dom([f, g])$, which contradicts the assumption that S is simple. Now assume that $h = [f, g] \circ c$ with $f, g, c \in R_1$. Then $S = rg(h) \subseteq E: = \{\xi | g(\xi) = 0\}$. This means: $\bar{E} \subseteq \bar{S}$. $\bar{E}$ is r.e. (even decidable) and infinite; otherwise, every ξ would have a terminal successor or a successor in a cycle and, hence, $dom([f, g])$ and $dom([f, g] \circ c) = S$ would be decidable, which contradicts the assumption that S is simple.

Remark The foregoing theorem and its proof is a good example demonstrating that to derive basic properties of the systems encountered in computer science one needs assumptions on the effectiveness of the components of automata that are not present in the concept of an autonomous discrete time system in general system theory. For instance, it is easy to represent any partial function h in the form $h = r \circ [f, g]$ with total r, f, g if we do not require r, f, g to be recursive.

2.2. THEOREM *Let $\rho \in R_1$. There exist $\Psi, \kappa \in R_1$, $\gamma \in R_2$ such that $\rho \circ [\Psi, \kappa] \circ \gamma$ is a Gödel numbering (i.e. ρ is a "possible" output coding for a universal automaton) iff ρ is onto and $\{\xi | (\exists \xi' < \xi)\ (\rho(\xi') = \rho(\xi))\}$ is infinite.*

Proof

Sufficiency That ρ has to be onto is obvious.

Define $\Psi: = \rho \circ [\Psi, \kappa] \circ \gamma$

$$A_\rho: = \{\xi | (\exists \xi' < \xi)(\rho(\xi') = \rho(\xi))\}$$

$$B_\rho: = A_\rho \cup \{\xi' | (\exists \xi \in A_\rho)(\rho(\xi') = \rho(\xi))\}.$$

Assume $|A_\rho| = m \in N$. Then $|B_\rho| \leq 2m$. The set $G: = \{\gamma(p, x) | \Psi(p, x)\uparrow\}$ is infinite because otherwise "$\Psi(p, x)\uparrow$" would be decidable. We choose a $\xi \in G$ that does not belong to B_ρ. There is no $\xi' \neq \xi$ such that $\rho(\xi') = \rho(\xi)$. Let $r: = \rho(\xi)$.

Then

$$(\forall p, x)(\Psi(p, x) \neq r) \qquad (1)$$

because if $r = \Psi(p, x)$ there would be a t such that

$$r = \rho\Psi^{(t)}\gamma(p, x),\ \Psi^{(t)}\gamma(p, x) = \xi$$

and

$$\kappa(\xi) = \kappa\Psi^{(t)}\gamma(p, x) = 0,$$

which contradicts $\rho[\Psi, \kappa](\xi)\uparrow$. Assertion (1), however, is not possible for the Gödel numbering Ψ.

Necessity We again define $A_\rho: = \{\xi | (\exists \xi' < \xi)\ (\rho(\xi') = \rho(\xi))\}$. A_ρ is decidable. Since A_ρ is infinite we can find two functions δ, δ' in R_1 such that δ maps N bijectively onto A_ρ and $\delta'\delta(x) = x$. Let $f(r): = (\min y)\ (\rho(y) = r)$. Since ρ is onto, $f \in R_1$. Moreover, $rg(f) = \bar{A}_\rho$. We now take $\rho', \Psi', \kappa' \in R_1, \gamma' \in R_2$ such that $\Psi: = \rho' \circ [\Psi', \kappa'] \circ \gamma'$ is a Gödel numbering and define

$$\gamma(p, x): = \delta\gamma'(p, x) \quad (\text{i.e. } \gamma(p, x) \in A_\rho)$$

$$\Psi(\xi): = \begin{cases} \delta\Psi'\delta'(\xi) & \text{if } \kappa'\delta'(\xi) \neq 0 \\ f\rho'\delta'(\xi) & \text{otherwise} \end{cases}$$

$$\kappa(\xi): = \begin{cases} 0 & \text{if } \xi \in \bar{A}_\rho \quad (\text{i.e. } \xi \in rg(f)) \\ 1 & \text{if } \xi \in A_\rho \quad (\text{i.e. } \xi \in rg(\delta)). \end{cases}$$

By the construction of γ, Ψ, κ it is clear that $\Psi, \kappa \in R_1, \gamma \in R_2$, and $\rho \circ [\Psi, \kappa] \circ \gamma = \Psi$.

2.3. THEOREM *Let $\gamma \in R_2$. There exist $\rho, \Psi, \kappa \in R_1$ such that $\rho \circ [\Psi, \kappa] \circ \gamma$ is a Gödel numbering iff there exists a $\beta \in R_1$ such that $\gamma(\beta(p), x)$ is a 1–1 function of p and x (i.e. γ is a "possible" input coding for a universal automaton iff γ is 1–1 at least in an effectively accessible "cylinder" in $N \times N$).*

Proof If $\rho \circ [\Psi, \kappa] \circ \gamma$ is a Gödel numbering, then there is a $\beta \in R_1$ such that $\langle p, x \rangle = \rho[\Psi, \kappa]\gamma(\beta(p), x)$. The function $\gamma(\beta(p), x)$ must be 1–1, otherwise $\langle p, x \rangle$ would not be a pairing function.

To see the converse implication we need two lemmas.

2.4. LEMMA *Let $\gamma \in R_2$, $\beta \in R_1$ and $\gamma(\beta(p), x))$ be 1–1.*

Then there is a $\beta' \in R_1$ such that $\gamma(\beta'(p), x)$ is 1–1 and the complement of $\gamma(\beta'(N), N)$ contains an infinite decidable subset.

Proof Define $\beta'(p): = \beta(2p)$ and $\beta''(p): = \beta(2p + 1)$. Then $M: = \gamma(\beta''(N), N)$ is contained in the complement of $\gamma(\beta'(N), N)$. Since M is recursively enumerable and infinite, it contains an infinite decidable subset.

2.5. LEMMA *Let $f \in P_1$ and $E \subseteq N$ be an infinite, decidable set. Then there exist $\rho, \Psi, \kappa \in R_1$ such that $(\forall \xi \in \bar{E})\,(f(\xi) = \rho[\Psi, \kappa](\xi))$.*

Proof Let ρ', Ψ', κ', $\gamma' \in R_1$ be such that $f = \rho' \circ [\Psi', \kappa'] \circ \gamma'$. Since E is decidable, a well-known construction yields δ, $\delta' \in R_1$ such that δ maps N onto E and $\delta'\delta(x) = x$. We define

$$\Psi(\xi) := \begin{cases} \delta\gamma'(\xi) & \text{if } \xi \in \bar{E} \\ \delta\Psi'\delta'(\xi) & \text{if } \xi \in E \end{cases}$$

$$\kappa(\xi) := \begin{cases} 1 & \text{if } \xi \in \bar{E} \\ \kappa'\delta'(\xi) & \text{if } \xi \in E \end{cases}$$

$$\rho(\xi) := \rho'\delta'(\xi).$$

The assertion is now clear from the definition of Ψ, κ, ρ.

Proof of Theorem 2.3 (continued)

By Lemma 2.4 we choose a $\beta' \in R_1$ and an infinite decidable set $E \subseteq N$ such that $\gamma(\beta'(p), x)$ is 1–1 and

$$E \text{ is contained in the complement of } \gamma(\beta'(N), N). \tag{1}$$

Take a Gödel numbering Ψ. The function η defined by

$$\eta(z) := \gamma(\beta'(z._1), z._2)$$

is a 1–1, recursive function. Of course,

$$\gamma(\beta'(p), x) = \eta\langle p, x\rangle. \tag{2}$$

We can find a (partial!) function $\eta' \in P_1$ such that

$$\eta'\eta(z) = z. \tag{3}$$

By Lemma 2.5 we know that

$$(\forall \xi \in \bar{E})(\Psi(\eta'(\xi)._1, \eta'(\xi)._2) = \rho[\Psi, \kappa](\xi)) \tag{4}$$

for some $\rho, \Psi, \kappa \in R_1$. We have

$$\begin{aligned} \Psi(p,x) &= \Psi(\langle p,x\rangle ._1, \langle p,x\rangle ._2) \\ &= \Psi((\eta'\eta\langle p,x\rangle)._1, (\eta'n\langle p,x\rangle)._2) \\ &\quad \text{by (3)} \\ &= \Psi((\eta'\gamma(\beta'(p),x))._1, (\eta'\gamma(\beta'(p),x))._2) \\ &\quad \text{by (2)} \\ &= \rho[\Psi,\kappa]\gamma(\beta'(p),x) \\ &\quad \text{by (1) and (4),} \end{aligned}$$

i.e. $\rho \circ [\Psi, \kappa] \circ \gamma$ is a Gödel numbering because the Gödel numbering Ψ can be recursively translated into it.

ACKNOWLEDGEMENTS

Professors Albrecht and Helmberg (University of Innsbruck) made it possible for the two authors in their departments to co-operate on this paper. To them we express our sincere gratitude. The Austrian Fonds zur Förderung der wissenschaftlichen Forschung (Fund for the Promotion of Scientific Research) supported part of our work under Project No. 2661.

REFERENCES

1. H. Rogers Jr., *Theory of Recursive Functions and Effective Computability*. McGraw Hill, New York, 1967.
2. P. Wegner, "Operational semantics of programming languages". *ACM SIGPLAN Notices*, 7, No. 1, 1972, (Proceedings of an ACM Conference on Proving Assertions about Programs, Las Cruces, New Mexico, January 6–7, 1972) pp. 128–141.
3. B. Buchberger, "On certain decompositions of Gödel numberings and the semantics of programming languages". In: *Lecture Notes in Computer Science*, 5, Springer, Berlin, 1974, (Proceedings of the International Symposium on Theoretical Programming, August 1972, Novosibirsk, edited by A. Ershov and V. A. Nepomniaschy), pp. 152–171.
4. R. C. Roehrkasse, L. Chiaraviglio, "Automata and digital computers". *International Journal of General Systems*, **1**, No. 3, 1974, pp. 189–196.
5. M. Blum, "A machine-independent theory of the complexity of recursive functions". *Journal of the ACM*, 14, No. 2, 1967, pp. 322–336.
6. J. P. Cleave, "Creative functions". *Zeitschrift für mathemathische Logik und Grundlagen der Mathematik*, **7**, No. 3, 1961, pp. 205–212.
7. Yu. L. Ershov, "Theorie der Numerierungen I", *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, **19**, 1973, pp. 289–388.
8. R. M. Friedberg, "Three theorems on recursive enumeration". *Journal of Symbolic Logic*, **23**, No. 3, 1958, pp. 309–316.
9. R. Smullyan, *Theory of formal systems*. Annals of mathematics studies, No. 47. Princeton, N.J., 1961.
10. J. C. Shepherdson, "Machine configuration and word problems of given degree of unsolvability". *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, **11**, 1965, pp. 149–175.
11. J. P. Cleave, Combinatorial systems I. Cylindrical problems". *Journal of Computer and System Sciences*, **6**, No. 3, 1972, pp. 254–266.