SOLVING SEQUENTIAL CONDITIONS BY FINITE-STATE STRATEGIES[1]

BY

J. RICHARD BUCHI AND LAWRENCE H. LANDWEBER

Our main purpose is to present an algorithm which decides whether or not a condition $\mathfrak{C}(X, Y)$ stated in sequential calculus admits a finite automata solution, and produces one if it exists. This solves a problem stated in [4] and contains, as a very special case, the answer to Case 4 left open in [6]. In an equally appealing form the result can be restated in the terminology of [7], [10], [15]: Every ω-game definable in sequential calculus is determined. Moreover the player who has a winning strategy, in fact, has a winning finite-state strategy, that is one which can effectively be played in a strong sense. The main proof, that of the central Theorem 1, will be presented at the end. We begin with a discussion of its consequences.

1. **Conditions on sequential operators.** Let $\mathfrak{C}(X, Y)$ be a condition (i.e., binary relation) on ω-sequences $X = X0, X1, X2, \ldots$ and $Y = Y0, Y1, Y2, \ldots$ of members of the finite sets I and J respectively. Let $Y = \mathscr{A}(X)$ be an operator which maps I-sequences into J-sequences. We will say that the operator $\mathscr{A}$ *solves the condition* $\mathfrak{C}(X, Y)$ *for* Y or that $\mathscr{A}$ is a *solution of* $\mathfrak{C}$ *for* Y, if $(\forall X)\mathfrak{C}(X, \mathscr{A}(X))$ or equivalently,

$$(\forall XY) \cdot Y = \mathscr{A}(X) \supset \mathfrak{C}(X, Y). \tag{1}$$

If no further requirement is imposed on solutions, then the axiom of choice states: $(\forall X)(\exists Y)\mathfrak{C}(X, Y)$ is the solvability condition of $\mathfrak{C}$ for Y. The solvability question becomes more interesting if one requires the solution $\mathscr{A}$ to be continuous in the sense of the *natural Cantor topology* on the set of all ω-sequences over the alphabets I and J. Let I^* denote the set of all finite sequences (words) over I. The members of I^* form a tree if all words wa, $a \in I$ are taken as direct successors of $w \in I^*$. ω-sequences over I are represented by infinite paths through the tree. Let U_w be the set of all those paths X which contain w (as an initial segment). The finite unions $U_{w_1} \cup \cdots \cup U_{w_n}$ are then the open-closed (clopen) sets of the totally disconnected space of all I-sequences. An operator $Y = \mathscr{A}(X)$ is *continuous* if it may be given in the form,

$$Yt = \Phi(\bar{X}(\phi t)) \tag{2}$$

whereby $\bar{X}z$ stands for the word $X0, \ldots, Xz$, ϕ is a map from ω into ω and Φ maps I^* into J.

Received by the editors October 19, 1967 and, in revised form, April 15, 1968.

(1) This research was sponsored by the National Science Foundation Grant No. GJ-120. The main result was announced in [13].

Among the continuous operators there are those for which the entries in the sequence $Y=\mathscr{A}(X)$ can in fact be computed, if sufficient information about the entries in X is provided. The *recursive operators* (RO) are those presentable in form (2), whereby both ϕ and Φ are recursive.

A particularly simple class of recursive operators are the *finite automata operators* (FAO), that is those operators which may be presented in the form,

$$Z0 = H[X0], \qquad Zt' = L[Xt', Zt], \qquad YT = W[Zt], \tag{3}$$

where $t'=t+1$. Here Z varies over ω-sequences from a finite set K. H, L and W are functions from I into K, $I\times K$ into K, and K into J. A system $\langle K, H, L, W\rangle$ is called a *finite automaton* with *input states* I, *output states* J, and (internal) *states* K. Finite automata were first studied by Kleene [12]. Also see [3], [5], and [16]. Besides being recursive, FAO's are deterministic in the sense that the state of Y at time t can be calculated without anticipating future states of the input X. More precisely, a continuous operator (2) is *deterministic* (DO) if $\phi t \leqq t$. I.e., if it can be given in the form,

$$Yt = \Phi(\bar{X}t). \tag{4}$$

Thus we use the term deterministic in the sense familiar from physics.

Note that a DO is continuous but need not be recursive. A FAO is a recursive deterministic operator (RDO). Furthermore, one easily proves: The DO given by (4) is a FAO if and only if the right congruence $u\sim v$ on words, defined by $(\forall w)\Phi(uw)=\Phi(vw)$, has finite index. This explains in just which way a finite automaton is limited in its ability to memorize the input history $\bar{X}t$ at time t. To be a FAO is a very strong requirement on a RDO.

The operator (2) might be called *h-shift* in case $\phi t=0$ for $t<h$, $\phi t=t-h$ for $h\leqq t$. The deterministic operators now appear as 0-shift $\supseteq$ 1-shift $\supseteq$ 2-shift $\supseteq \cdots$. In particular (4) is a 0-shift operator and a 1-shift operator is one of form

$$Yt = \Phi(\bar{X}(t-1)) \tag{4'}$$

whereby $\bar{X}(-1)$ stands for the empty word. Furthermore, the FAO defined by (3) is a 0-shift FAO, while a 1-shift FAO can always be presented in the form

$$Z0 = c, \qquad Zt' = L[Xt, Zt], \qquad Yt = W[Zt], \tag{3'}$$

whereby $c\in K$ is called the *initial state* of the 1-shift automaton $\langle K, c, L, W\rangle$.

DEFINITION 1. A condition $\mathfrak{C}$ is called *determined* if either there exists a 0-shift deterministic solution $Y=\mathscr{A}(X)$ of $\mathfrak{C}(X, Y)$ for Y or else there exists a 1-shift deterministic solution $X=\mathscr{B}(Y)$ of $\sim\mathfrak{C}(X, Y)$ for X.

It is interesting to contemplate this notion in the context of the Cantor topology; say for example, if $\mathfrak{C}$ is a Borel set in the product of the two spaces. This is studied in a game theoretic context in [7], [10], and [15]. If $\mathfrak{C}$ is determined, it either contains the graph of a continuous function $Y=\mathscr{A}(X)$, or else $\sim\mathfrak{C}$ contains the graph of a continuous function $X=\mathscr{B}(Y)$.

LEMMA 1. *Let $\mathfrak{C}$ be an arbitrary condition. There cannot both exist a 0-shift deterministic solution $Y=\mathscr{A}(X)$ of $\mathfrak{C}(X, Y)$ for Y and a 1-shift deterministic solution $X=\mathscr{B}(Y)$ of $\sim\mathfrak{C}(X, Y)$ for X.*

Proof. Suppose $Y=\mathscr{A}(X)$ is a 0-shift solution of $\mathfrak{C}$ for Y, given by $Yt=\Phi(\bar{X}t)$, and $X=\mathscr{B}(Y)$ is a 1-shift solution of $\sim\mathfrak{C}$ for X, given by $Xt=\Psi'(\bar{Y}(t-1))$. The system of equations $Yt=\Phi(\bar{X}t)$, $Xt=\Psi'(\bar{Y}(t-1))$ can be viewed as a simultaneous course-of-value induction, defining a pair X_0, Y_0 satisfying both equations, for all values of t. But then $Y_0=\mathscr{A}(X_0)$ and $X_0=\mathscr{B}(Y_0)$. Therefore, if $\mathscr{A}$ solves $\mathfrak{C}$ for Y and $\mathscr{B}$ solves $\sim\mathfrak{C}$ for X, we have $\mathfrak{C}(X_0, Y_0)$ and $\sim\mathfrak{C}(X_0, Y_0)$, which is contradictory. Q.E.D.

2. **Finite-state conditions, the ω-behavior of finite automata.** Let $Z=\mathscr{E}(X, Y)$ be a FAO from ω-sequences on $I\times J$ into ω-sequences on S, given by the recursions,

$$Z0 = s_0, \qquad Zt' = H[Xt, Yt, Zt]. \tag{5}$$

Here s_0 is a member of S and H maps $I\times J\times S$ into S. Furthermore, let U be a class of subsets of S, called the *output condition*. Let $\sup Z$ denote the set of all states taken infinitely often by Z. I.e.,

$$s\in \sup Z \,.\equiv.\, (\forall x)(\exists t)[x \leqq t \wedge Zt = s]. \tag{6}$$

DEFINITION 2. The *ω-behavior of* $\langle S, s_0, H, U\rangle$ (of the FAO $\mathscr{E}$ with output condition U) is the relation $\mathfrak{C}(X, Y)$ which holds for X and Y if $Z=\mathscr{E}(X, Y)$ satisfies $\sup Z\in U$. I.e.,

$$\mathfrak{C}(X, Y) \,.\equiv.\, (\exists Z)[Z0 = s_0 \wedge (\forall t)Zt' = H[Xt, Yt, Zt] \wedge \sup Z\in U]. \tag{7}$$

By a *finite-state condition* we mean one which is the ω-behavior of some FAO with output condition. Our basic result may be stated thus:

THEOREM 1. *Every finite-state condition $\mathfrak{C}(X, Y)$ is determined. Moreover, either there is a 0-shift FAO which solves $\mathfrak{C}$ for Y or else there is a 1-shift FAO which solves $\sim\mathfrak{C}$ for X.*

The proof is contained in §5. Actually we obtain there a constructive version of Theorem 1. In §3 we discuss a game theoretic form of this theorem which was conjectured by McNaughton. The purpose of §4 is to show that a surprisingly wide class of formulas $\mathfrak{C}$ in fact define finite-state conditions. We thereby extend the applicability of Theorem 1. An important step in this extension is provided by a recent result of McNaughton [14], which can truly be called the fundamental lemma of finite automata behavior. It can be stated as follows.

In place of the initial state s_0 we assume a set of initial states $K\subseteq S$. In place of the function H from $I\times J\times S$ into S we consider a relation L on $I\times J\times S\times S$. An

ω-sequence Z on S is called a *transition sequence* of the *transition system* (sometimes called nondeterministic finite automaton) $\langle S, K, L\rangle$ if

$$(5') \qquad K[Z0], \qquad L[Xt, Yt, Zt, Zt'], \quad \text{for all } t.$$

The notion of ω-behavior naturally generalizes to transition systems. Namely,

DEFINITION 2'. The *ω-behavior of the transition system $\langle S, K, L, U\rangle$ with output condition* is the relation $\mathfrak{C}(X, Y)$ which holds for X and Y if there is a transition sequence Z such that $\sup Z \in U$. I.e.,

$$(7') \qquad \mathfrak{C}(X, Y) \,.\equiv.\, (\exists Z)[K[Z0] \wedge (\forall t)L[Xt, Yt, Zt, Zt'] \wedge \sup Z \in U].$$

The fundamental lemma now states that ω-behaviors of transition systems are still finite state conditions. More precisely,

FUNDAMENTAL LEMMA (MCNAUGHTON). *To every transition system with output condition $\boldsymbol{L}=\langle S, K, L, U\rangle$ on the input states $I\times J$ one can effectively construct a finite automaton with output condition $\boldsymbol{H}=\langle S', s_0, H, W\rangle$ on $I\times J$, such that $\boldsymbol{L}$ and $\boldsymbol{H}$ have the same ω-behavior.*

Thus Theorem 1 remains true if $\mathfrak{C}(X, Y)$ is the ω-behavior of a transition system. A further extension is discussed in §4.

3. **ω-games and sequential conditions.** McNaughton has observed a close relationship between the notion of a deterministic solution of a condition $\mathfrak{C}(X, Y)$ and that of a winning strategy in purely combinatorial ω-games studied in the literature [7], [10], and [15]. While this game terminology is not really needed for our purpose, it puts both the solvability problems of automata theory and game theory into a wider context, and adds appealing flavors to each. For example, the notion of determinateness (Definition 1) is very natural in terms of games, but did not arrive independently in automata theory. Indeed we could have avoided all reference to solutions of $\sim\mathfrak{C}(X, Y)$ for X, in a presentation of our solvability algorithm. But this would clearly be hiding important information.

A condition $\mathfrak{C}(X, Y)$ can be viewed as a game for two, player I and player J. Intuitively, a play of the game $\mathfrak{C}(X, Y)$ goes as follows. At any time $t=0, 1, 2, \ldots$ player I makes a move Xt by selecting a member of I. Then player J follows up with a move Yt from J. The play $\langle X, Y\rangle$ is completed when all ω moves $X0, Y0, X1, Y1, \ldots$ have been made. Player J wins if $\mathfrak{C}(X, Y)$, else player I wins. It is intended that at time t, player I has complete information about all previous moves $\bar{Y}(t-1)$ of his opponent, and player J has complete information about all previous moves $\bar{X}t$ of his opponent. More rigorously this can be stated thus:

DEFINITION 3. *A strategy for player I (for player J)*, in a game $\mathfrak{C}(X, Y)$, is a deterministic 1-shift operator $X=\mathscr{B}(Y)$ (deterministic 0-shift operator $Y=\mathscr{A}(X)$). If $\langle\mathscr{B}, \mathscr{A}\rangle$ is a pair of such strategies, then the *play* $\langle\mathscr{B}, \mathscr{A}\rangle$ produces the pair $\langle X_0, Y_0\rangle$ such that $\mathscr{A}(X_0)=Y_0$ and $\mathscr{B}(Y_0)=X_0$. The strategy $\mathscr{A}$ (of player J) *beats* the strategy $\mathscr{B}$ (of player I) in case $\mathfrak{C}(X_0, Y_0)$. Otherwise $\mathscr{B}$ beats $\mathscr{A}$. *A winning strategy* for either player is one which beats all strategies of the opponent.

That the play $\langle \mathscr{B}, \mathscr{A} \rangle$ exists has been pointed out in the proof of Lemma 1. We leave it to the reader to verify

LEMMA 2. *An operator* $Y=\mathscr{A}(X)$ $(X=\mathscr{B}(Y))$ *is a winning strategy for player J (player I) in the game* $\mathfrak{C}(X, Y)$ *if and only if it is a deterministic* 0*-shift* (1*-shift*) *solution of the condition* $\mathfrak{C}$ *for* Y $(\sim\mathfrak{C}$ *for* $X)$.

Thus Lemma 1 asserts the intuitively obvious fact that in no game $\mathfrak{C}$ can both players possess a winning strategy. Furthermore, the condition $\mathfrak{C}(X, Y)$ is determined (Definition 1) just in case the game $\mathfrak{C}$ is determined in the sense that one of the two players possesses a winning strategy.

If $\mathfrak{C}(X, Y)$ is called a *finite-state game* in case it is the ω-behavior of a finite automata operator of form (5) with an output U, then Theorem 1 takes the following form.

THEOREM 1′. *Every finite-state game is determined. Moreover, the player who has a winning strategy in fact has one which can be executed by a finite automaton.*

We leave it to the reader to make up a particular finite-state game and to meditate about the sense in which such a game can actually be played. We also suggest that he review the results of §4 in game terminology.

We would like to emphasize here that the second stronger part of Theorem 1′ is critical for our solvability algorithm (§4). This second part is also a new kind of result in game theory. More generally, the following type of game problem is naturally suggested by automata theory. Given a class of games G: (1) can one effectively decide, for any $\mathfrak{C} \in G$, which player has a winning strategy? (2) Just how simple winning strategies do exist for games in G? For example, is there a recursive or even a finite automata winning strategy for $\mathfrak{C} \in G$? This general problem was considered in [17].

We suggest that the arithmetic hierarchy [11] provides more natural choices of G (in connection with the above questions), than does the classical Borel hierarchy considered in the literature [7], [10] and [15]. To state a more concrete question we ask:

PROBLEM. For any $\forall_3$-game is there a winning strategy in the arithmetic hierarchy of operators? If yes, how high do they occur in the hierarchy?

Here $\forall_3$ stands for the class of all $\mathfrak{C}(X, Y)$ which are of the form

$$(\forall x)(\exists y)(\forall z)B(X, Y, x, y, z),$$

whereby B is recursive. Note that $\forall_3$ is contained in $F_{\sigma\delta}$ of the Borel hierarchy over the product of the natural Cantor spaces of I-sequences and J-sequences (since $B(X, Y, x, y, z)$ is open and closed for fixed x, y and z). Hence $\forall_3$ games are determined as a consequence of the following result of Davis [7].

(*) All $F_{\sigma\delta}$ games are determined.

It is easy to show, using the axiom of choice, that there is a $\mathfrak{C}(X, Y)$ which is not determined [10]. However, it is not known whether all $F_{\sigma\delta\sigma}$ or even all $\forall_4$ games are determined.

For comparison with our stronger proof of the full Theorem 1′, we end this digression into game theory with a proof, using (*), that all finite state games are determined. In fact Theorem 2 below is somewhat stronger.

Call $\mathfrak{C}(X, Y)$ a *continuous*-sup-*condition* (*recursive*-sup-*condition*) if it is of the form, $\sup Z \in U$ if $Z = \mathscr{E}(X, Y)$, whereby $\mathscr{E}$ is a continuous operator (recursive operator). I.e.,

$$(8) \qquad \mathfrak{C}(X, Y) \,.\equiv.\, (\exists Z)[(\forall t)Zt = \Phi(\bar{X}(\phi t), \bar{Y}(\phi t)) \wedge \sup Z \in U]$$

where Φ and ϕ are arbitrary (recursive) functions. Note that ω-behaviors (i.e., finite-state games) are recursive-sup.

LEMMA 3. *Every continuous*-sup-*condition* (*recursive*-sup-*condition*) $\mathfrak{C}$ *is in the Boolean algebra over* F_σ (*over* $\exists_2$).

Proof. Assume $\mathfrak{C}$ is given by (8), but drop the second argument Y to avoid notational complexity. Using the definition of sup (6) it follows that

$$\mathfrak{C}(X) \,.\equiv.\, \bigvee_{B \in U} \Big[(\exists y)(\forall t)[y \leqq t \supset \Phi(\bar{X}(\phi t)) \in B] \wedge \bigwedge_{s \in B} (\forall y)(\exists t)[y \leqq t \wedge \Phi(\bar{X}(\phi t)) = s]\Big].$$

Note that U and its members B are finite sets, so that $\mathfrak{C}(X)$ is a Boolean combination of expressions of the form $(\exists y)(\forall t)M(X, y, t)$. The expressions $M(X, y, t)$, namely $[y > t \vee \Phi(\bar{X}(\phi t)) \in B]$ or $[y > t \vee \Phi(\bar{X}(\phi t)) \neq s]$ (for various values of B and s), denote clopen sets for fixed y and t (recursive relations in case Φ and ϕ are recursive). This is true because $M(X, y, t)$ implies $M(X^*, y, t)$ whenever $\bar{X}^*(\phi(t)) = \bar{X}(\phi(t))$. Consequently each $(\exists y)(\forall t)M(X, y, t)$ denotes an F_σ (an $\exists_2$) so that $\mathfrak{C}$ is a Boolean combination of F_σ's (of $\exists_2$'s). Q.E.D.

THEOREM 2. *Every continuous*-sup-*game* (*recursive*-sup-*game*) $\mathfrak{C}(X, Y)$ *is determined.*

The proof is obvious from (*) and Lemma 3. We have not investigated whether Davis' proof of (*) can be analyzed to yield further information in case one assumes $\mathfrak{C}(X, Y)$ to be recursive-sup (or even an ω-behavior). At any rate, if $\mathfrak{C}(X, Y)$ is an ω-behavior our Theorem 1′ strengthens Theorem 2.

It seems unlikely that there is a presentation for recursive-sup-conditions which admits a method for deciding which of the players has a winning strategy. Note that our Theorem 6 states the existence for sequential conditions.

PROBLEM. Is it true that for every recursive-sup-game either of the players has a winning strategy which is arithmetical? If yes, how high does it occur?

4. **A solvability-synthesis algorithm for sequential calculus.** Our concern here is not so much to determine solutions for particular conditions. We rather ask for

algorithms which for a class CL of conditions determine solvability questions with respect to a class OP of operators. Such algorithms are discussed in the literature [2], [5], [6], [8], [9], [18] and [19]. We will restate some known results and show what our basic Theorem 1 provides.

Let CL be an interpreted formalism (called the condition language) containing formulas $\mathfrak{C}(X, Y)$ denoting relations between ω-sequences. Let OP be a class of operators. A *solvability algorithm for* CL with respect to OP is an effective procedure which applies to any $\mathfrak{C}(X, Y) \in$ CL and tells whether or not $\mathfrak{C}$ admits a solution $\mathscr{A} \in$ OP for Y. In case the members of OP are finitely presentable (as is a FAO by a finite automaton and a RDO by a Turing machine computing Φ), one may ask for a *partial synthesis algorithm* which for any $\mathfrak{C}(X, Y) \in$ CL constructs a presentation of a solution $\mathscr{A} \in$ OP, if a solution exists, and a *solution algorithm* which, given a $\mathfrak{C}(X, Y) \in$ CL and a presentation of some $\mathscr{A} \in$ OP, decides whether or not $\mathscr{A}$ solves $\mathfrak{C}$ for Y.

In [4] *sequential calculus* (SC) is considered as a natural candidate for a condition language for FAO. SC is the monadic second-order theory of the successor function on natural numbers. That is SC is the interpreted formalism which includes the first order theory of $\langle \omega, 0, ' \rangle$ and quantification over monadic predicate variables ranging over sets of natural numbers. Note that a subset X of ω (i.e., predicate on ω) may also be interpreted as an ω-sequence of members of $\{T, F\}$, and a finite sequence $X = \langle X_1, \ldots, X_k \rangle$ is an ω-sequence of members of the set $\{T, F\}^k$. Thus, a formula $\mathfrak{C}(X, Y)$ of SC with free predicate variables $X = \langle X_1, \ldots, X_h \rangle$ and $Y = \langle Y_1, \ldots, Y_k \rangle$ denotes a condition on ω-sequences over $I = \{T, F\}^h$ and $J = \{T, F\}^k$. Other finite I and J can be handled by coding their members as sequences of the truth-values T, F.

In [4] a method for deciding truth of sentences in SC was presented. Let $Y = \mathscr{A}(X)$ be a FAO given by (3). By appropriate coding of the automaton $\langle K, H, L, W \rangle$ one can construct a formula $\mathfrak{F}(X, Y) \in$ SC of form,

$$(\exists Z_1 \cdots Z_n) \,.\, Z_1 0 \equiv H_1(0) \wedge \cdots \wedge Z_n 0 \equiv H_n(0)$$
$$\wedge\ (\forall t)[Z_1 t' \equiv L_1(t) \wedge \cdots \wedge Z_n t' \equiv L_n(t')]$$
$$\wedge\ (\forall t)[Y_1 t \equiv W_1(t) \wedge \cdots \wedge Y_k t \equiv W_k(t)]$$

(whereby the H, L, W's are propositional formulas in the atomic parts $X_1 0, \ldots, X_h 0$, $Y_1 0, \ldots, Y_k 0$, $Z_1 0, \ldots, Z_n 0$, $X_1 t', \ldots, X_h t'$, $Y_1 t', \ldots, Y_k t'$, $Z_1 t, \ldots, Z_n t$), such that $\mathfrak{F}(X, Y)$ means $Y = \mathscr{A}(X)$. The assertion, $\mathscr{A}$ solves $\mathfrak{C}(X, Y) \in$ SC for Y, where $\mathscr{A}$ is a FAO can therefore be stated as a sentence of SC. Hence,

THEOREM 3. *There is a solution algorithm and a partial synthesis algorithm for SC with respect to FAO.*

A partial synthesis algorithm is available because all finite automata can be effectively enumerated, and one after the other checked as to whether it solves a proposed condition $\mathfrak{C}(X, Y)$ stated in SC. For a very small fragment of SC a

solvability algorithm was found [5] and improved by Wright. This result can be extended to cover conditions of SC of the form

$$(\text{predicate prefix on } Z)\,.\ H[Z0] \wedge (\forall t)L[Xt,\ Yt,\ Zt,\ Zt']$$

(unpublished). It is easy to see [4] that addition of an existential conjunct $(\exists t)M[Zt]$ to such formulas yields all conditions $\mathfrak{C}(X, Y)$ expressible in SC. However, even for very special formulas including both kinds of individual quantifiers the problem of finding a solvability algorithm was left open in [6], and seemed rather hopeless at that time. It is only by using McNaughton's fundamental lemma and our Theorem 1 that we are now able to give a solvability algorithm for all of SC.

The main definability result of [4] can be restated thus: To every formula $\mathfrak{C}(X, Y)$ of SC one can construct a transition system $\langle S, K, L, U\rangle$ with output condition whose ω-behavior is (the relation defined by) $\mathfrak{C}$. Thus by the fundamental lemma

THEOREM 4. *To every formula $\mathfrak{C}(X, Y)$ of SC one can effectively construct a finite automaton with output $\langle S, s_0, H, U\rangle$ whose ω-behavior is (the relation defined by) $\mathfrak{C}$.*

Conversely, the ω-behavior of every finite automaton can be defined in SC. In fact (7) with sup Z replaced by its definition (6), up to coding, yields such a definition (as $x \leqq y$ is definable in SC by $(\forall Z)[Zy \wedge (\forall t)[Zt' \supset Zt] \supset Zx]$). Also note that the fundamental lemma yields another proof of the critical Lemma 9 of [4] (as explained in [14]), which does not make use of Ramsey's Theorem.

Because of Theorem 4 we can extend Theorem 1 to

THEOREM 5. *Every condition $\mathfrak{C}(X, Y)$ definable in SC is determined. In fact, either there is a 0-shift FAO which solves $\mathfrak{C}$ for Y or else there is a 1-shift FAO which solves $\sim\mathfrak{C}$ for X.*

Because of Lemma 1 we have

COROLLARY. *If $\mathfrak{C}(X, Y)$ in SC has a deterministic solution for Y then it has a FAO solution for Y.*

The corollary generalizes the statement: If $\mathfrak{C}(Y)$ in SC holds for some Y, then it holds for an ultimately periodic Y. Just note that a FAO solution of $\mathfrak{C}(Y)$ for Y is an input free automaton.

THEOREM 6. *There is an algorithm which for any $\mathfrak{C}(X, Y)$ of SC, decides whether $\mathfrak{C}(X, Y)$ is deterministically solvable for Y, and either* (1) *produces a 0-shift FAO solution of $\mathfrak{C}(X, Y)$ for Y if $\mathfrak{C}$ is deterministically solvable for Y or* (2) *produces a 1-shift FAO solution of $\sim\mathfrak{C}(X, Y)$ for X if $\mathfrak{C}$ is not deterministically solvable for Y.*

Proof. ALGORITHM 1. Systematically list all 0-shift FAO $Y=\mathscr{A}(X)$ and all 1-shift FAO $X=\mathscr{B}(Y)$. Check whether $\mathscr{A}$ solves $\mathfrak{C}$ for Y or $\mathscr{B}$ solves $\sim\mathfrak{C}$ for X using the algorithm of Theorem 3. By Theorem 5, eventually a solution of $\mathfrak{C}$ for Y or a solution of $\sim\mathfrak{C}$ for X will be found.

ALGORITHM 2. Use the algorithm of Theorem 4 to put $\mathfrak{C}(X, Y)$ in finite state form. Then use the method described in §5.

Note that there is a solvability algorithm for SC with respect to DO, which is also a solvability algorithm for SC with respect to FAO (Theorem 6). However, while there is a solution algorithm with respect to FAO (Theorem 3), there is no solution algorithm for SC with respect to RDO. For example, let $\mathfrak{C}(X, Y)$ be $(\exists y)(\forall z)[y<z \supset \sim Yz]$. Let $Y=\mathscr{A}_Q(X)$ be the RDO defined by

$$Yt = T \quad \text{if } t \in Q,$$
$$= F \quad \text{if } t \notin Q,$$

whereby Q is a recursive set. $\mathscr{A}_Q$ solves $\mathfrak{C}$ for Y if and only if Q is finite. Hence a solution algorithm for SC with respect to RDO would, given an index for Q, decide whether it is finite. It is well known that such a method does not exist.

We have not seriously investigated whether the algorithms of Theorems 3 and 6 can be improved to a point of usefulness in the deisgn of sequential circuits. As they include conversion of propositional formulas into normal form, it seems that presently available computing equipment could not carry a significant part of our algorithms. Nevertheless, our solution automata of §5, like the construction of [14], provide examples of strictly finite devices which accomplish surprisingly intricate tasks.

The fundamental lemma can be extended to α-behaviors, for any countable ordinal α. This leads to a decision method for the monadic second-order theory of $\langle \alpha, < \rangle$ (see [1]). We hope to present elsewhere a corresponding extension of Theorem 6 from ω to any countable ordinal α.

$\mathfrak{C}(X, Y)$ admits an h-shift solution for Y, if and only if,

$$\mathfrak{C}_h(X, Y) : (\exists Z) \,.\, \mathfrak{C}(Z, Y) \wedge (\forall t) Zt = X(t+h)$$

has a 0-shift solution for Y. Thus, for any fixed h, Theorem 6 yields a solvability algorithm for SC with respect to h-shift DO's and FAO's. Note that any $(h+1)$-shift recursion is also an h-shift recursion. This suggests

PROBLEM. Can one algorithmically determine whether or not for a condition $\mathfrak{C}(X, Y)$ stated in SC there exists an h such that $\mathfrak{C}$ admits an h-shift, but no $(h+1)$-shift solution for Y?

5. **Solving finite state conditions.** We will present here our main proof, that of Theorem 1. Therefore, throughout this section $\mathfrak{C}(X, Y)$ will be the ω-behavior, with respect to U, of the FAO given by

$$Z0 = s_0, \qquad Zt' = H[Xt, Yt, Zt]. \tag{9}$$

Let I, J, S be the finite sets of states of X, Y, Z respectively, so that U is a class of subsets of S. We recall that $\mathfrak{C}(X, Y)$ stands for sup $Z \in U$, whereby Z is given by (9).

Our proof is outlined as follows. In section (a) we will construct a subset $R_i[\ ]$ of the set of states S, (the significance of the brackets will be made clear below)

such that if $s_0 \in R_l[\ ]$ then $\mathfrak{C}(X, Y)$ is solvable for Y by a 0-shift FAO, and if $s_0 \notin R_l[\ ]$ then $\sim\mathfrak{C}(X, Y)$ is solvable for X by a 1-shift FAO. Thus, $s_0 \in R_l[\ ]$ is the condition of solvability of $\mathfrak{C}(X, Y)$ for Y. The case $s_0 \in R_l[\ ]$ is treated in section (b), where we will present a 0-shift FAO $Y=\mathscr{A}(X)$ which solves $\mathfrak{C}$ for Y. The case $s_0 \notin R_l[\ ]$ is treated in section (c), where a 1-shift FAO $X=\mathscr{B}(Y)$ is presented which solves $\sim\mathfrak{C}$ for X.

(a) *Definition of* $R_l[\ ]$. For each $A \in U$ choose a cyclic permutation of its members. For simplicity of notation we denote the value of this permutation at $s \in A$ by $A(s)$. The crucial construction is that of the sets $R_k[A_1, s_1, \ldots, A_n, s_n]$, $P_k[A_1, s_1, \ldots, A_n, s_n]$, and $Q_k[A_1, s_1, \ldots, A_n, s_n]$, whereby $n \geqq 0$, $A_1 \supset \cdots \supset A_n$ range over strictly decreasing chains in U and $s_1, \ldots, s_n$ range over members of $A_1, \ldots, A_n$, respectively. Note that $B \subset A$ will always mean "B is properly contained in A." These sets are defined simultaneously by the following induction on $k=0, 1, 2, \ldots$.

$$s \in R_0[A_1, s_1, \ldots, A_n, s_n] \,.\equiv.\, \text{false},$$

$$s \in P_k[A_1, s_1, \ldots, A_n, s_n] \,.\equiv.\, s \in R_k[\ ] \vee s \in A_1 \cap R_k[A_1, s_1] \vee \cdots \vee s \in A_n \cap R_k[A_1, s_1, \ldots, A_n, s_n],$$

$$(10)\quad s \in Q_k[A_1, s_1, \ldots, A_n, s_n] \,.\equiv.\, \bigvee_B .\, B \in U \wedge s \in B \subset A_n \wedge \bigwedge_{u \in B} u \in R_k[A_1, s_1, \ldots, A_n, s_n, B, B(u)],$$

$$s \in R_{k+1}[A_1, s_1, \ldots, A_n, s_n] \,.\equiv.\, \bigwedge_{x \in I} \bigvee_{y \in J} H[x, y, s] \in \{s_1, \ldots, s_n\} \cup P_k[A_1, s_1, \ldots, A_n, s_n] \cup Q_k[A_1, s_1, \ldots, A_n, s_n].$$

Note that n is bounded by the length of maximal chains in U. If $n=0$ we use the notations $R_k[\ ]$, $P_k[\ ]$, $Q_k[\ ]$.

Caution! In interpreting (10) for the case $n=0$ the occurrence of A_n (in the expression $s \in B \subset A_n$) is to be suppressed. A similar remark goes for all future occurrences of A_0. Also, the set $\{s_1, \ldots, s_n\}$ is the empty set if $n=0$.

By induction on k, one easily shows that $R_k[v] \subseteq R_{k+1}[v]$, $P_k[v] \subseteq P_{k+1}[v]$, $Q_k[v] \subseteq Q_{k+1}[v]$, for all arguments $v=[A_1, s_1, \ldots, A_n, s_n]$. We include the induction step for the R sets and leave it to the reader to complete the proof.

Assume $R_k[v] \subseteq R_{k+1}[v]$, $P_k[v] \subseteq P_{k+1}[v]$ and $Q_k[v] \subseteq Q_{k+1}[v]$ for all v. Let $v=[A_1, s_1, \ldots, A_n, s_n]$. Then

$$s \in R_{k+1}[v] \,.\equiv.\, \bigwedge_{x \in I} \bigvee_{y \in J} H[x, y, s] \in \{s_1, \ldots, s_n\} \cup P_k[v] \cup Q_k[v]$$

$$\text{(Ind. hyp.)} \qquad .\supset.\, \bigwedge_{x \in I} \bigvee_{y \in J} H[x, y, s] \in \{s_1, \ldots, s_n\} \cup P_{k+1}[v] \cup Q_{k+1}[v]$$

$$.\supset.\, s \in R_{k+2}[v].$$

Because all $R_k[v]$, $P_k[v]$ and $Q_k[v]$ are subsets of the finite set S, and there are

but a finite number of v's, it follows that there is a number k such that, for all v, $R_k[v]=R_{k+1}[v]$, $P_k[v]=P_{k+1}[v]$, and $Q_k[v]=Q_{k+1}[v]$. Accordingly we define l

(11) l is the first number such that, $R_{l-1}[v]=R_l[v]$, $P_{l-1}[v]=P_l[v]$, and $Q_{l-1}[v]=Q_l[v]$, for all $v=[A_1, s_1, \ldots, A_n, s_n]$, $A_1 \supset \cdots \supset A_n$, $n \geqq 0$.

From (10) and (11) we obtain

$$s \notin R_l[A_1, s_1, \ldots, A_n, s_n] \cdot \equiv \cdot \bigvee_{x \in I} \bigwedge_{y \in J} H[x, y, s] \notin \{s_1, \ldots, s_n\} \cup P_l[A_1, s_1, \ldots, A_n, s_n] \cup Q_l[A_1, s_1, \ldots, A_n, s_n],$$

$$(12) \quad s \notin Q_l[A_1, s_1, \ldots, A_n, s_n] \cdot \equiv \cdot \bigwedge_{B} [B \in U \wedge s \in B \subset A_n] \supset \bigvee_{u \in B} u \notin R_l[A_1, s_1, \ldots, A_n, s_n, B, B(u)],$$

$$s \notin P_l[A_1, s_1, \ldots, A_n, s_n] \cdot \equiv \cdot s \notin R_l[\ \ ] \wedge s \notin A_1 \cap R_1[A_1, s_1] \wedge \cdots \wedge s \notin A_n \cap R_l[A_1, s_1, \ldots, A_n, s_n].$$

(b) *The case* $s_0 \in R_l[\ \ ]$. Choose a linear order of the members of J and U. An expression $(\mu y)E(y)$ denotes the first member y of J, in the chosen order, which satisfies $E(y)$, if it exists. We will now display a 0-shift FAO $Y=\mathscr{A}(X)$, and prove that it solves $\mathfrak{C}(X, Y)$ for Y.

In the sequel X, Y, Z, k denote ω-sequences over the sets I, J, S, $\{0, \ldots, l\}$, respectively. V denotes ω-sequences of elements of form $[A_1, s_1, h_1, \ldots, A_n, s_n, h_n]$, whereby $n \geqq 0$, $A_1 \supset \cdots \supset A_n$ is a chain of members of U, $s_1 \in A_1, \ldots, s_n \in A_n$ and $l > h_1 > \cdots > h_n$. Consider the following formulas (13), which define Y, Z, V and k recursively from X,

$$Z0 = s_0, \qquad V0 = [\ \ ], \qquad k0 = l.$$

Assume now that $Vt=[A_1, s_1, h_1, \ldots, A_n, s_n, h_n]$. Then,

$$Yt = (\mu y) \cdot H[Xt, y, Zt] \in \{s_1, \ldots, s_n\} \cup P_{kt-1}[A_1, s_1, \ldots, A_n, s_n] \cup Q_{kt-1}[A_1, s_1, \ldots, A_n, s_n],$$

$$Zt' = H[Xt, Yt, Zt].$$

(α) If $Zt' \in \{s_1, \ldots, s_n\}$, let i be the first such that $Zt' = s_i$. Then,
$Vt' = [A_1, s_1, h_1, \ldots, A_i, A_i(s_i), h_i]$
$kt' = h_1$.

(13) (β) If $Zt' \in P_{kt-1}[A_1, s_1, \ldots, A_n, s_n]$ but not (α), let j be the first such that $Zt' \in A_j \cap R_{kt-1}[A_1, s_1, \ldots, A_j, s_j]$ ($Zt' \in R_{kt-1}[\ \ ]$ if $j = 0$). Then,
$Vt' = [A_1, s_1, h_1, \ldots, A_j, s_j, h_j]$
$kt' = kt-1$.

(γ) If $Zt' \in Q_{kt-1}[A_1, s_1, \ldots, A_n, s_n]$ but neither (α) nor (β), let B be the first in the chosen order of U such that, $Zt' \in B \subset A_n$ and $\bigwedge_{u \in B} u \in R_{kt-1}[A_1, s_1, \ldots, A_n, s_n, B, B(u)]$. Then,
$Vt' = [A_1, s_1, h_1, \ldots, A_n, s_n, h_n, B, B(Zt'), kt-1]$
$kt' = kt-1$.

Note also the formulas (14).

$$\text{If } Vt=[A_1, s_1, h_1, \ldots, A_n, s_n, h_n], \text{ then}$$

$$Zt \in R_{kt}[A_1, s_1, \ldots, A_n, s_n], \quad Zt \in A_n \qquad (\text{if } n \neq 0),$$

$$A_1 \supset \cdots \supset A_n, \quad s_i \in A_i \in U, \quad l > h_1 > \cdots > h_n \geqq kt > 0, \tag{14}$$

$$\bigwedge_{u \in A_l} u \in R_{h_l}[A_1, s_1, \ldots, A_i, A_i(u)].$$

Because we are dealing with the case $s_0 \in R_l[\ ]$, the values $Z0$, $V0$, $k0$ given by (13) clearly satisfy (14). Assume inductively that (14) holds for t, and Xt is arbitrary. Using (10) it follows that there is a $y \in J$ such that

$$H[Xt, y, Zt] \in \{s_1, \ldots, s_n\} \cup P_{kt-1}[A_1, s_1, \ldots, A_n, s_n] \cup Q_{kt-1}[A_1, s_1, \ldots, A_n, s_n],$$

and therefore Yt exists as described by (13), and so do Zt', Vt', kt', in all cases (α), (β), (γ). Furthermore, one easily checks that these values Zt', Vt', kt' satisfy (14) with t replaced by t'. Thus, the formulas (13) constitute a recursive definition of Y, Z, V, k from X, and furthermore, (13) implies (14).

Let $Y=\mathscr{A}(X)$ be the operator from I-sequences to J-sequences, given by (13). Then $\mathscr{A}$ clearly is deterministic and recursive. Furthermore, because of (14), the auxiliaries Z, V, k in the recursion (13) are finite valued. In fact it is easy to modify (13) so that it is of form (3). Therefore, $\mathscr{A}$ is a 0-shift FAO. It remains to show that $\mathscr{A}$ solves $\mathfrak{E}$ for Y, i.e., that (1) holds.

Note that a copy of (9) is built into the definition (13) of $\mathscr{A}$. As a consequence the assertion (1) is tantamount to the assertion: For any X, Y, Z, V, k, (13) implies $\sup Z \in U$. The remainder of section (b) constitutes a proof of this.

Assume that (13), and therefore (14), holds for X, Y, Z, V, k. From (13) one easily sees that $Vt_1=[\ ]$ and $t_1<t_2$ implies $kt_1>kt_2$. (Prove by induction on t that if $V_t=[A_1, s_1, h_1, \ldots, A_n, s_n, h_n]$ for $t>t_1$, then (a) $kt<kt_1$ and (b) $h_i<kt_1$ for $i=1, \ldots, n$. This is obviously true for t_1+1. Assume it is true for t and observe that (α), (β) and (γ) of (13) preserve (a) and (b).) Therefore by (14), there can be but finitely many t such that $Vt=[\ ]$. Accordingly there is a t_1 such that, $Vt\neq[\ ]$ for all $t \geqq t_1$, i.e., if $t \geqq t_1$ then Vt is of form $[A_1, s_1, h_1, \ldots, A_n, s_n, h_n]$ with level $n \geqq 1$.

As the level n of Vt is bounded by the lengths of chains in U (see (14)), some level $n \geqq 1$ must occur infinitely often. Let m be the smallest of these. Then, $m \geqq 1$ and there is a t_2 such that for all $t \geqq t_2$ the level of Vt is $\geqq m$. Thus we have,

(15) If $t \geqq t_2$ then $Vt=[A_1, s_1, k_1, \ldots, A_m, s_m, k_m, \ldots, A_n, s_n, k_n]$ whereby $n \geqq m$. Furthermore, $n=m$ occurs for infinitely many times t.

It follows from (15) that for $t'>t_2$ only the cases (α) $i \geqq m$, (β) $j \geqq m$, and (γ) $n \geqq m$ of (13) can occur. Consequently, for $t \geqq t_2$ the entry A_m (and all previous entries) in Vt remains constant. By (14) it follows that $Zt \in A_m$ for $t \geqq t_2$, $A_m \in U$. Thus $\sup Z \subseteq A_m \in U$.

Suppose the case (α) $i=m$ occurred for only finitely many t. Then there would

be a $t_3 > t_2$ such that for $t \geqq t_3$ only the cases (α) $i > m$, (β) $j \geqq m$, (γ) $n \geqq m$ could occur. Inspection of (13, 14) shows that then also the case (β) $j = m$ could occur only for finitely many $t \geqq t_3$ because each application of (β) $j = m$ followed by 0 or more steps (γ) $n \geqq m$, (α) $i > m$, (β) $j > m$ followed by (β) $j = m$ lowers the value of k. (This is shown as follows: Assume (β) $j = m$ is used to obtain Vt and $V(t+c)$, $c \geqq 1$ but no $V(t+\bar{c})$ for $\bar{c} < c$. Prove by induction on $\bar{c} \leqq c$ that if $V(t+\bar{c}) = [A_1, s_1, h_1, \ldots, A_m, s_m, h_m, A_{m+1}, \ldots, h_{m+n}]$, then (a) $h_{m+j} < kt$, $j = 1, \ldots, n$ and (b) $k(t+\bar{c}) < kt$.) Thus both cases (α) $i = m$ and (β) $j = m$ would occur only finitely often. This contradicts the second part of (15). Therefore the case (α) $i = m$ must occur infinitely often.

Let $t_3 < t_4 < t_5 < \cdots$ be the infinitely many consecutive places $t > t_2$ where (α) $i = m$ is used. It clearly follows that $Zt_4 = A_m(Zt_3), Zt_5 = A_m(Zt_4), Zt_6 = A_m(Zt_5), \ldots$. Because $s \to A_m(s)$ was chosen to be a cyclic permutation of A_m, it follows that Z will keep taking any value in A_m. Thus, $\sup Z \supseteq A_m$. Together with a former result, this yields $\sup Z = A_m \in U$. Q.E.D.

(c) *The case* $s_0 \notin R_l[\ ]$. Choose a linear order of I. The expression $(\mu x)E(x)$ denotes the first x in I such that $E(x)$, if it exists. We will now display a 1-shift FAO $X = \mathscr{B}(Y)$, and prove that it solves $\sim\mathfrak{C}(X, Y)$ for X.

In the sequel X, Y, Z denote ω-sequences over the sets I, J, S. V denotes ω-sequences over elements of form $[A_1, s_1, \ldots, A_n, s_n]$, whereby $A_1 \supset \cdots \supset A_n$ is a chain of members of U and $s_1 \in A_1, \ldots, s_n \in A_n$. W denotes ω-sequences of chains of subsets of S. Consider the following formulas (16), which define V, W, X, and Z recursively from Y.

$$Z0 = s_0, \qquad W0 = \{\{s_0\}\}, \qquad V0 = [\ ].$$

Assume that $Vt = [A_1, s_1, \ldots, A_n, s_n]$. Then,

$$Xt = (\mu x) \bigwedge_{y \in J} H[x, y, Zt] \notin \{s_1, \ldots, s_n\} UP_l[A_1, s_1, \ldots, A_n, s_n] \cup Q_l[A_1, s_1, \ldots, A_n, s_n].$$

$$Zt' = H[Xt, Yt, Zt].$$

$$Wt' = \{B \cup \{Zt'\};\ B \in Wt \vee B \text{ empty}\}.$$

(16) (α) If $\bigvee_B . B \in U \cap Wt' \wedge Zt' \in B \wedge \bigvee_{0 \leqq i \leqq n} [[(0 < i < n \wedge A_i \supset B \supset A_{i+1})] \vee (i = n \wedge B \subset A_n) \vee (i = 0 \wedge B \supset A_1)] \wedge Zt' \notin R_l[A_1, s_1, \ldots, A_i, s_i, B, B(Zt')]$, let B be the largest such. (Note that if $n = 0$, then the above reduces to $\bigvee_B B \in U \cap Wt' \wedge Zt' \in B \wedge Zt' \notin R_l[B, B(Zt')]$.)

$$Vt' = [A_1, s_1, \ldots, A_i, s_i, B, B(Zt')].$$

(β) If not (α), let i be such that $Zt' \in A_i$, $Zt' \notin A_{i+1}$ (A_{n+1} empty).

$$Vt' = [A_1, s_1, \ldots, A_i, s_i].$$

Note: $Vt' = [\ ]$, if $n = 0$.

Note also the formulas (17).

$$\text{If } Vt = [A_1, s_1, \ldots, A_n, s_n], \text{ then}$$

$$(17)\qquad \begin{aligned} &Zt \notin R_l[A_1, s_1, \ldots, A_n, s_n], \qquad A_1 \supset \cdots \supset A_n \text{ all in } U \cap Wt,\\ &Zt \in A_n \text{ (if } n \neq 0), \qquad s_1 \in A_1, \ldots, s_n \in A_n,\\ &Wt \text{ is a chain of subsets of } S. \end{aligned}$$

Because we are dealing with the case $s_0 \notin R_l[\ ]$, the values $Z0$, $P0$, $V0$ given by (16) satisfy (17) for $t=0$. Assume inductively that (17) holds for t and Yt is any member of J. By (12) it follows that Xt, Zt', Wt' as prescribed by (16) exists. Furthermore, Wt' is still a chain of subsets of S, and $Zt' \notin P_l[A_1, s_1, \ldots, A_n, s_n]$. By (12) it therefore follows that, in cases (β), (17) holds for t replaced by t'. The same can easily be checked in case Vt' is calculated by (α). The preceding argument shows that (16) constitutes a recursive definition of Z, W, V, X from Y, and that (16) implies (17).

Let $X=\mathscr{B}(Y)$ be the operator from I-sequences to J-sequences, given by (16). Then $\mathscr{B}$ clearly is a 1-shift deterministic operator. Furthermore, the auxiliaries Z, W, V take values in finite sets (see (17)). In fact, the recursion (16) is easily modified to the form (3′). Thus, $X=\mathscr{B}(Y)$ is a 1-shift FAO. To terminate the proof of Theorem 1, it remains to be shown that $\mathscr{B}$ solves $\sim\mathfrak{C}(X, Y)$ for X, i.e., that $X=\mathscr{B}(Y)$ and (9) imply $\sup Z \notin U$.

Note that the recursion (9) is built into the definition (16) of $X=\mathscr{B}(Y)$. As a consequence, "$\mathscr{B}$ solves $\sim\mathfrak{C}(X, Y)$ for X" is tantamount to the assertion: (16) implies $\sup Z \notin U$. The remainder of this section constitutes a proof of this, in the form: (16) and $\sup Z \in U$ yields a contradiction.

For the sequel assume that (16), and therefore (17), holds for X, Y, Z, W, V. Furthermore, assume $\sup Z = D \in U$. It follows that there is a t_1 such that

$$(18)\qquad \begin{aligned} &t \geqq t_1 \supset Zt \in D,\\ &u \in D \supset (\exists t)[t \geqq a \wedge Zt = u], \text{ for any time } a. \end{aligned}$$

From (16) one clearly sees that the chain Wt consists of all sets $\{Z0, \ldots, Zt\}$, $\{Z1, \ldots, Zt\}, \ldots, \{Z(t-1), Zt\}, \{Zt\}$. It follows from (18), and $D \in U$ that there is a time $t_2 \geqq t_1$, such that

$$(19)\qquad t \geqq t_2 \supset D \in U \cap Wt.$$

Let $Vt=[A_1, s_1, \ldots, A_n, s_n]$. From (16), $Zt' \notin Q_l[A_1, s_1, \ldots, A_n, s_n]$. Because of (12) and $B \in Wt' \supset Zt' \in B$ this yields

$$[B \in U \cap Wt' \wedge (n = 0 \vee A_n \supset B)] \supset \bigvee_{u \in B} u \notin R_l[A_1, s_1, \ldots, A_n, s_n, B, B(u)].$$

This and (19) yield,

$$[t \geqq t_2 \wedge (n = 0 \vee A_n \supset D)] \supset \bigvee_{u \in D} u \notin R_l[A_1, s_1, \ldots, A_n, s_n, D, D(u)].$$

Because of (18) this yields,

$$(20)\qquad \begin{aligned} [t \geqq t_2 \wedge Vt = [A_1, s_1, \ldots, A_n, s_n] \wedge (n = 0 \vee A_n \supset D)]\\ \supset (\exists a)[a \geqq t \wedge Za' \notin R_l[A_1, s_1, \ldots, A_n, s_n, D, D(Za')]]. \end{aligned}$$

Define the quasi-order $\prec$ on chains $A_1 \supset \cdots \supset A_p$ $(p \geqq 0)$ of members of U by:

$$[B_1, \ldots, B_q] \prec [A_1, \ldots, A_p] \cdot \equiv \cdot \bigvee_{1 \leqq i \leqq q, p} \Big[\bigwedge_{1 \leqq j \leqq i} A_j = B_j \wedge A_i \supset B_i \Big] \vee \Big[\bigwedge_{1 \leqq j \leqq q} A_j = B_j \wedge p > q \Big].$$

By the *principal part* of the chain $A_1 \supset \cdots \supset A_n$ (of $[A_1, s_1, \ldots, A_n, s_n]$) we mean the chain $A_1 \supset \cdots \supset A_p$ (the sequence $[A_1, s_1, \ldots, A_p, s_p]$) whereby p is the largest i such that $A_i \supseteq D$, or $p = 0$ if there is no such i. Note that $A_p \supseteq D \supset A_{p+1}$ if $1 \leqq p < n \neq 0$, $D \subseteq A_p$ if $p = n$, and $p = 0$ if $n = 0$.

Let $Vt = [A_1, s_1, \ldots, A_n, s_n]$, let $[A_1, s_1, \ldots, A_p, s_p]$ be the principal part of Vt. Inspection of (16) shows that the principal part of Vt' will be equal or larger (in the sense of $\prec$), except if $(\beta)_{i<p}$ comes to use. If $t \geqq t_2$, so that by (18), $Zt' \in D \subseteq A_p$, $(\beta)_{i<p}$ cannot come to use. Therefore, for $t \geqq t_2$ the principal part of Vt either stays equal or increases. Because $\prec$ is a quasi-order on a finite set, there must be a $t_3 \geqq t_2$ such that the principal part of Vt remains constant from t_3 on. I.e., there are $m \geqq 0$, $\bar{A}_1, \bar{s}_1, \ldots, \bar{A}_m, \bar{s}_m$ such that $m = 0$ or $\bar{A}_m \supseteq D$ and,

(21) if $t \geqq t_3$ then Vt is of form $[\bar{A}_1, \bar{s}_1, \ldots, \bar{A}_m, \bar{s}_m, \ldots, A_n, s_n]$ whereby $n = m$ or $D \supset A_{m+1} \supset \cdots \supset A_n$.

Assume that, for all $t \geqq t_3$, Vt were of form $[\bar{A}_1, \bar{s}_1, \ldots, \bar{A}_m, \bar{s}_m, A_{m+1}(t), s_{m+1}(t), \ldots]$. By (16), it follows that $A_{m+1}(t') \supseteq A_{m+1}(t)$, for all $t \geqq t_3$. Thus, there would have to be a $c \geqq t_3$ such that $A_{m+1}(t)$ remained constant, say $= A$, from c on. By (21) $D \supset A$, so that by (18) there would exist a $d \geqq c$, $Zd' \notin A$. But $Vd = [\bar{A}_1, \bar{s}_1, \ldots, \bar{A}_m, \bar{s}_m, A, \ldots]$, so that the case $(\beta)_{i=m}$ of (16) would come to play. As a result, $Vd' = [\bar{A}_1, \bar{s}_1, \ldots, \bar{A}_m, \bar{s}_m]$. This is contradictory to the assumption, so that there must be a $t_4 \geqq t_3$ such that,

$$Vt_4 = [\bar{A}_1, \bar{s}_1, \ldots, \bar{A}_m, \bar{s}_m]. \tag{22}$$

Assume $\bar{A}_m \supset D$, or $m = 0$. By (20) and (22) there would be an $a \geqq t_4$ such that $Za' \notin R_l[\bar{A}_1, \bar{s}_1, \ldots, \bar{A}_m, \bar{s}_m, D, D(Za')]$. By (21), $Va = [\bar{A}_1, s_1, \ldots, \bar{A}_m, \bar{s}_m]$, or $Va = [\bar{A}_1, \bar{s}_1, \ldots, \bar{A}_m, \bar{s}_m, A, \ldots]$ and $D \supset A$. By (19), $D \in U \cap Wa'$. Thus, D is a possible value for B in (α) of (16). Therefore, (α) for some $B \supseteq D$ would come to use for calculating Va'. The result would be an entry $B \notin \{\bar{A}_1, \ldots, \bar{A}_m\}$, $B \supseteq D$ in Va'. This contradicts (21). Therefore,

$$m \geqq 1 \wedge \bar{A}_m = D. \tag{23}$$

From (16) it clearly follows that $Zt' \neq \bar{s}_m$ if $Vt' = [\bar{A}_1, \bar{s}_1, \ldots, \bar{A}_m, \bar{s}_m, \ldots]$. Therefore, by (21) and (17), $Zt' \in \bar{A}_m$ and $Zt' \neq \bar{s}_m \in \bar{A}_m$ for any $t \geqq t_3$. It follows that $\bar{A}_m$ can not be sup Z, i.e., $\bar{A}_m \neq D$. Together with (23) this yields the contradiction, ending the proof of Theorem 1.

The reader will easily find various modifications of our recursions (13) which also define DO's which solve $\mathfrak{C}(X, Y)$ for Y in case $s_0 \in R_l[\ ]$. For example, such a recursion may keep $Zt' \in R_l[\ ]$ up to a fixed time h and from there on act like

(13). More generally, the time h, at which the forcing of $\sup Z$ into U is actually started, may be made to depend on the input X, and may be set and reset, deterministically depending on X. We have not investigated the following question.

PROBLEM. Modify the recursions (13) to a schema with parameters which, by proper additional specifications for the parameters, will yield any given deterministic operator which solves $\mathfrak{C}(X, Y)$ for Y. Do the same for (16) and solutions of $\mathfrak{C}(X, Y)$ for X.

To accomplish this it might be necessary to make (more basic) changes in the definition (10) of the sets $R_k[A_1, s_1, \ldots, A_n, s_n]$, which would make our proofs less intricate.

BIBLIOGRAPHY

1. J. R. Buchi, *Decision methods in the theory of ordinals*, Bull. Amer. Math. Soc. **71** (1965), 767–770.

2. J. R. Buchi, C. E. Elgot and J. B. Wright, *The nonexistence of certain algorithms of finite automata theory*, Notices Amer. Math. Soc. **5** (1958), 98.

3. J. R. Buchi, *Weak second-order arithmetic and finite automata*, Z. Math. Logik Grundlagen Math. **6** (1960), 66–92.

4. ———, *On a decision method in restricted second order arithmetic*, Proc. Internat. Congr. Logic, Method. and Philos. Sci. 1960, Stanford Univ. Press, Stanford, Calif., 1962.

5. A. Church, *Application of recursive arithmetic to the problem of circuit synthesis*, Summaries of Talks Presented at the Summer Institute for Symbolic Logic, Cornell Univ. 1957, 2nd ed; Princeton, N. J., 1960, 3–50.

6. ———, *Logic arithmetic and automata*, Proc. Internat. Congr. Math. 1963, Almqvist and Wiksells, Uppsala, 1963.

7. M. Davis, "Infinite games of perfect information" in *Advances in game theory*, Princeton Univ. Press, Princeton, N. J., 1964, 85–101.

8. C. C. Elgot, *Decision problems of finite automata design and related arithmetics*, Trans. Amer. Math. Soc. **98** (1961), 21–51.

9. J. Friedman, *Some results in Church's restricted recursive arithmetic*, J. Symbolic Logic **22** (1957), 337–342.

10. D. Gale and F. M. Stewart, "Infinite games with perfect information" in *Contributions to the theory of games*, Vol. II, Princeton Univ. Press, Princeton, N. J., 1953, 245–266.

11. S. C. Kleene, *Arithmetical predicates and function quantifiers*, Trans. Amer. Math. Soc. **79** (1955), 312–340.

12. ———, "Representation of events in nerve nets and finite automata" in *Automata studies*, Princeton Univ. Press, Princeton, N. J., 1956, 3–41.

13. L. H. Landweber, *Finite state games—A solvability algorithm for restricted second-order arithmetic*, Notices Amer. Math. Soc. **14** (1967), 129–130.

14. R. McNaughton, *Testing and generating infinite sequences by a finite automaton*, Information and Control **9** (1966), 521–530.

15. J. Mycielski, S. Swierczkowski and A. Zieba, *On infinite positional games*, Bull. Acad. Polon. Sci. Cl. III **4** (1956), 485–488.

16. M. Rabin and D. Scott, *Finite automata and their decision problems*, IBM J. Res. Develop. **3** (1959), 114–125.

17. M. Rabin, "Effective computability of winning strategies" in *Contributions to the theory of games*, Vol. III, Princeton Univ. Press, Princeton, N. J., 1957, 147–157.

18. B. A. Trachtenbrot, *Synthesis of logic networks whose operators are described by means of single place predicate calculus*, Dokl. Akad. Nauk SSSR **118** (1958), 646–649.

19. ———, *Finite automata and the logic of single place predicates*, Dokl. Akad. Nauk SSSR **140** (1961), 320–323 = Soviet Math. Dokl. **2** (1961), 623–626.

20. J. R. Buchi and L. G. Landweber, *Definability in the monadic second-order theory of successor*, J. Symbolic Logic **34** (1969).

PURDUE UNIVERSITY,
LAFAYETTE, INDIANA
UNIVERSITY OF WISCONSIN,
MADISON, WISCONSIN