T H E U N I V E R S I T Y O F M I C H I G A N

COLLEGE OF LITERATURE, SCIENCE, AND THE ARTS

Department of Philosophy — Logic of Computers Group

Final Report

DESIGN ALGORITHMS IN AUTOMATA LANGUAGES

A. W. Burks
J. R. Büchi
C. C. Elgot
J. B. Wright

UMRI Project 2755

under contract with:

DEPARTMENT OF THE ARMY
OFFICE OF ORDNANCE RESEARCH
DETROIT ORDNANCE DISTRICT
CONTRACT NO. DA-20-018-ORD-16971
DETROIT, MICHIGAN

administered by:

THE UNIVERSITY OF MICHIGAN RESEARCH INSTITUTE ANN ARBOR

July 1960

Other agencies of the Federal Government assisted in the support of much of this research. These agencies are the Office of Naval Research, the U. S. Army Signal Corps, and the National Science Foundation.

INTRODUCTION

Research under Contract No. Da-20-018-ORD-16971 began in the spring of 1958 and continued into June, 1960.

The details of the principal research are indicated in items 3 through 9 of this report. In particular, attention is directed to the two technical reports, Decision Problems of Finite Automata Design and Related Arithmetics, by Calvin Elgot, and Weak Second-Order Arithmetic and Finite Automata, by J. R. Buchi, which have already been distributed to the Office of Ordnance Research, and to On a Problem of Tarski by J. R. Buchi, which is a part of this final report.

Research has also been conducted in additional areas ancillary to this principal research area. In the technical report Lectures on Switching and Automata Theory by Calvin Elgot (distributed to the Office of Ordnance Research) the automata theory field is reviewed from simple switching theory up through the sequential machine theory of Kleene and Moore. In Sequence Generators and Digital Computers by A. W. Burks and J. B. Wright (included as item 10 of this final report), the theory of a broad class of finite automata is investigated, and numerous algorithms for the analysis and synthesis of these automata are indicated.

SUMMARY

The principal goal of our research has been to obtain algorithms which would permit the mechanizability of the processes of computer design. Attention was focused on the following design algorithms.

1. Solution algorithm: Given an automaton and given a behavior condition, to determine whether or not the automaton satisfies the condition.

2. Solvability algorithm: Given a behavior condition, to determine whether or not an automaton exists satisfying the condition.

3. Synthesis algorithm: Given a behavior condition, to produce the specifications for an automaton satisfying the condition (providing such an automaton exists).

The first goal of research was to ascertain the characteristics of a formal language suitable for setting down of automata structure descriptions, and automata behavior condition descriptions.

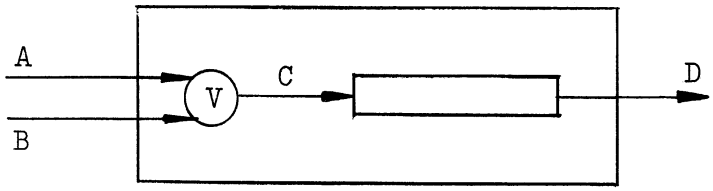
(The difference between an automaton and a condition is as follows: An automaton expression gives all the relationships existing between every switching and every delay element throughout the machine. A condition expression gives only the input to output relationship, i.e., only the conditions the designer wishes to have satisfied, with no reference to the relationships between the internal junctions by means of which the action is effected.)

In the case of the automata structure descriptions, it was apparent that the language to be employed should be a monadic predicate calculus where the predicates would denote junctions of switching or delay elements, and the individuals would denote instants of time. This same language could also be employed to express behavior conditions. For example,

$$A(0) \vee B(0) \supset D(1)$$

is a condition that possibly is satisfied by one or more automata. If we arrive at an expression for an automaton which might satisfy this condition, then the Solution algorithm could be applied to the automaton and to the condition to see if this is in fact the case.

Suppose it is suggested that the automaton pictured (below) satisfies the given behavior conditions: $A(0) \vee B(0) \supset D(1)$



This automaton expressed in our formal language is: $[A(0) \vee B(0) \equiv C(0)]$ and $[C(0) \equiv D(1)]$. (it should be noted that the condition expression omits mention of internal junction C, while the automaton expression makes explicit the presence and function of internal junction C.)

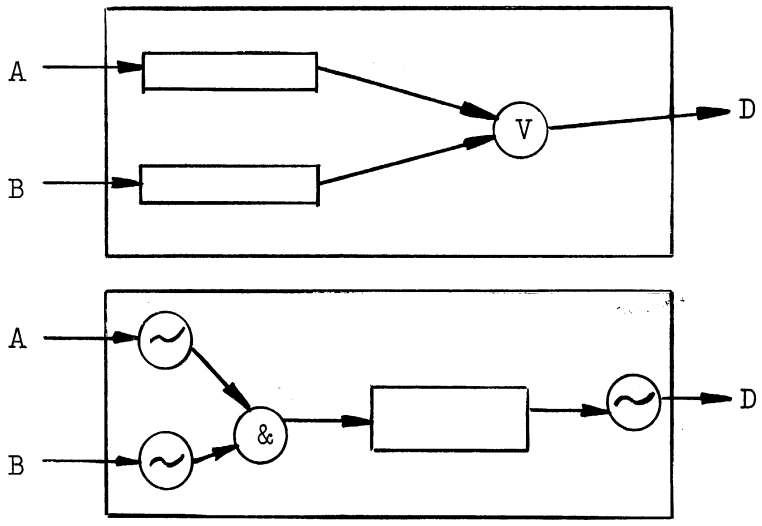
The Solution algorithm will judge the truth or falsity of the assertion that the given automaton implies the given behavior, i.e., the truth of falsity of the following expression:

$$\{[A(0) \vee B(0) \equiv C(0)] \wedge [C(0) \equiv D(1)]\} \supset \{[A(0) \vee B(0) \subset D(1)]\}$$

Now if there is a decision prodecure for the class of sentences of the formal language in which this expression is couched (a decision procedure is a method of mechanically determining, for any expression of a formal language, whether the expression is true or false), it then follows that we can mechanically determine whether the automaton satisfies the condition by merely ascertaining the truth or falsehood of this particular assertion by means of the general decision procedure for the language.

Thus it is evident that there may be a close connection between the decidability of the formal language chosen to express automata and conditions, and the existence or lack of it of our three automata design algorithms. We have seen that the existence of a Solution algorithm follows directly from the existence of a general decision procedure for the language employed. The Synthesis algorithm is also readily obtained. It is closely related to the Solution algorithm. In the Synthesis algorithm we are given a condition and asked to find an automaton satisfying the condition, providing such an automaton exists. There are various methods available for generating systematically the expressions for all automata. Automata expressions are then substituted, one by one, into the Solution algorithm and tested to see whether the automaton satisfies the behavior. This process of generating automaton descriptions, substituting them in the Solution algorithm, and testing whether the conditions are satisfied continues until a suitable automaton satisfying the conditions is obtained.

Since many automata may satisfy the same conditions, additional criteria for choosing the "best" automaton can be brought in (including, of course, minmum number of states, or minimum number of certain elements). For instance, the two automata pictured below both satisfy the condition $(A(0) \vee B(0) \supset D(1)$ of our earlier example, as does the automaton used in the example.



In setting down conditions for automata, it is advantageous to have a large vocabulary at one's disposal. In the examples so far, the vocabulary has been restricted to monadic predicates and their individuals, and to the connectives of propositional calculus. It would make the application of the design algorithms considerably easier if the designer, in setting down his conditions, could employ such expressions as "all," "equals," "less than," "some,""successor of," etc.

As we have noted, however, the existence or nonexistence of our design algorithms is closely related to the question of the decidability of the language as a whole, and it seems generally to be the case that the stronger a language is (the larger the number of its primitive operations, etc.), the more likely it is that the language is undecidable. We therefore find our objectives conflicting: the easier it is for a designer to set down his conditions, the more likely it is that there can never be a mechanical method of generating the desired automaton.

The second major goal of our research then became: to test large numbers of formal languages to determine whether the desired automata design algorithms existed in them, seeking to find the strongest useful language for which the algorithms existed. Since the question of the existence or lack of it of the algorithms seemed to be closely related to the question of the existence or lack of it of a general decision procedure for expressions in the language, the large body of research results in formal decidability was thoroughly reviewed. Often it was found that the problem of decidability for a formal language was still open, and a program of examining such languages, and perhaps solving the decision problem for them was undertaken. Both positive and negative solutions of decision problems were obtained for many languages.

In April of 1960, Dr. J. Richard Büchi, Research Mathematician with the Logic of Computers Group, obtained a positive solution to the decision problem for an extremely useful language. The existence of the Solution and Synthesis algorithms follows immediately in the manner indicated earlier. The question of the existence or not of a Solvability alborithm in this language is still open. (It had been answered in the affirmative for several useful, though weaker, languages.)

The language for which Dr. Büchi obtained a decision procedure we shall call the "sequential calculus." It has the following characteristics: it allows all expressions compounded from individual variables ranging over natural numbers, predicate variables ranging over arbitrary sets of natural numbers, quantification (all, some, etc.,) over <u>both</u> individual and predicate variables, the connectives (or, and, not, etc.,) of propositional calculus, equality, less than, and successor.

An example of a condition expression in the sequential calculus is given below.

$$(\exists R)\ (\forall t)\ \Big[\Big\{ S(t) \equiv [I(t) \,\&\, \sim J(t) \,\&\, \sim R(t)] \vee [\sim I(t) \,\&\, \sim J(t) \,\&\, R(t)] \vee$$
$$[\sim I(t) \,\&\, J(t) \,\&\, \sim R(t)]\Big\} \;\&\; \Big\{R(t) \equiv [I(t') \,\&\, \sim J(t') \,\&\, \sim R(t')] \vee$$
$$[\sim I(t') \,\&\, J(t') \,\&\, \sim R(t')] \vee [\sim I(t') \,\&\, \sim J(t') \,\&\, R(t')] \vee [I(t) \,\&\, J(t) \,\&\, R(t)]\Big\}\Big] \;\&$$
$$\exists (Q)\ (\forall t) \Big\{ [Q(t) \supset \sim I(t)] \;\&\; (\forall x)\ (\exists t)\ [t > x \;\&\; Q(t)]\Big\}$$

This is a behavior condition for the addition of real numbers less than one, where the sum is also less than one. This condition cannot be realized by a finite automaton.

A description of a realizable finite automaton behavior is given on following page.

$$(\exists R)\ \{R(o) \equiv \mathcal{G}[I(o)]\ \&$$

$$(\forall t)\ [R(t') \equiv \mathcal{H}[I(t), I(t'), R(t)]\ \&$$

$$(\forall t)\ [U(t) \equiv \mathcal{L}[R(t), I(t)]\}$$

This is a finite automaton transformation from infinite input sequences to infinite output sequences. The transformation, successfully expressed here in the sequential calculus, cannot be expressed in many of the formal languages which have been proposed as useful in automata design.

Lectures on Switching and Automata Theory

(2755-2-T, January, 1959)

Calvin C. Elgot

ABSTRACT

These lectures include a study of two-terminal series-parallel relay contact networks, multi-terminal relay contact networks, and sequential networks. General, systematic techniques for the study of series-parallel networks have employed Boolean algebra or propositional calculus. A basic paper on multi-terminal relay contact nets makes use of matrices whose entries are elements of a Boolean algebra. More recent work makes use of the algebraic concepts: lattice, semi-group, group. The concept of ring when suitably specialized is intimately connected with Boolean algebras. This indicates the extent to which algebra is invading the mathematical theory of switching. Much recent work has employed more advanced aspects of logic than the propositional calculus. Indeed, the use of logic and its techniques has already produced significant results and promises still more fruitful ones.

We begin with some preliminary, simple, algebraic and logical concepts which will lead to a discussion of finite Boolean algebras, propositional calculus, and their application to switching theory.

The work of Lunts (which lists results only) is discussed in Section 6. The arguments make use of an important correlation between matrices of zeros and ones and finite binary relations. This correlation is useful in other work on switching.

Automata and sequential circuits are introduced in Section 7 and the connection with Burks-Wright logical nets is pointed out. Moore's theory and related work of Mealy and Ginsburg is expounded. The equivalence, in a certain reasonable sense, of finite automaton, as defined here and as defined in Moore, is indicated. Finite semi-groups are associated with automata in Section 7,9. In the case of permutation automaton (backwards deterministic), the associated seni-group is a group. An application is made of this association. Kleene's theory of regularity is explained. An alternative notion of regularity is defined and its relation to the primary concept is established. The discussion terminates with an example illustrating several of the main results of the sections on automata theory.

[These lectures were prepared as Technical Report 2755-2-T, January, 1959, and were distributed to the addresses of the official OOR distribution list.]

Decision Problems of Weak Second-Order Arithmetics and Finite Automata

(Preliminary Report, Part I)

J. Richard Büchi and Calvin C. Elgot

ABSTRACT

1. Let L_1 be the class of formulas constructed out of atomic formulas $x_i \in F_j$, $y_k = x_\ell + 1$, $x_i < y_j$, by means of propositional connectives and quantifiers, $\exists x_i$, $\exists F_j$. The individual variables range over the natural numbers, N. The second-order variables range over finite sets of natural numbers. Each formula $A[F_1, F_2, \ldots, F_r]$ in L_1 may be interpreted as representing a set (usually infinite) of finite sequences of r-tuples of zeros and ones as well as sets "isomorphic" to that set. Theorem. A set of finite sequences of objects drawn from a finite set is representable by a formula in L_1 if and only if it is representable by a finite automaton (cf. Kleene, Automata Studies, or Copi, Elgot, and Wright, J.A.C.M, April, 1958).

2. Corollary. The set of true sentences of L_1 is recursive. The corollary has been obtained by A. Ehrenfeucht and R. L. Vaught via Ehrenfeucht's theorem (unpublished), stating that the elementary theory of addition of ordinals is decidable.

3. Corollary. There are solution and synthesis algorithms relative to the class of all automata and L_1. (Cf. Büchi, Elgot, and Wright, these Notices, February, 1958, for definitions.)

4. Each formula $A[F_1 F_2, \ldots, F_r]$ in L_1 defines $R \subseteq N^r$ via: $F \longrightarrow \sum_{n \in F} 2^n$. Corollary. The elementary theory of R, e.g., $x + y = z$ (Presburger), is decidable.

[This paper was presented at the 20-23 January 1959 meeting of the American Mathematical Society, in Philadelphia. The abstract was published in Notices of the American Mathematical Society, Vol 5, No. 7, Dec., 1958, p.834.]

Decision Problems of Weak Second-Order Arithmetics and Finite Automata

(Preliminary Report, Part II)

Calvin C. Elgot

ABSTRACT

1. Let L_2 be the class of formulas constructed out of atomic formulas $x_i \in F_j$, $y_k = x_\ell + 1$, $z_k = x_i + y_j$, by means of propositional connectives and individual quantification $(\exists x_i)$ only. The variables are interpreted as in L_1. Theorem. The set of satisfiable formulas of L_2 is recursively enumerable but not recursive.

2. If the set variables are interpreted as ranging over periodic sets (sets whose characteristic function is ultimately periodic), the theorem still holds.

3. Corollary. While there are solution and synthesis algorithms relative to the class of input-free automata and L_2, there is no solvability algorithm.

4. If the formulas of L_2 are interpreted over the integers rather than the natural numbers, the theorem still holds.

Let $E_A(a,b)$ be the set of finite sequences uabv where $a \in A$, $b \in A$, u and v are finite sequences (possibly null) of elements of A, and A is finite. The class of automaton representable sets is the smallest class of sets containing the sets $E_A(a,b)$, a unit set consisting of a sequence of length one, and closed under symmetric difference, intersection, and projection. (The mapping induced on sets of finite sequences of elements of A giving a set of finite sequences of elements of B, by an arbitrary mapping from A into B, is called a projection.) This strengthens a result of Medvedev.

[This paper was prepared for the American Mathematical Society Meeting in Philadelphia, 20-23 January, 1959. The abstract was published in the Notices of the American Mathematical Society, Vol. 6, No 1, February 1959, p.48.]

Decision Problems of Finite Automata Design and Related Arithmetics

(Technical Report 2755-6-T, June 1959)

Calvin C. Elgot

ABSTRACT

Certain formal arithmetics may be employed as design languages for finite automata design conditions, the notion of automaton, and the notion of an automaton satisfying a condition are expressible in these arithmetics. An automaton saisfies a condition if a certain formula of the arithmetic is valid.

For certain arithmetics, algorithms are produced which enable one to decide

(1) whether a given automaton satisfies a given condition,

(2) whether an automaton exists satisfying a given condition (and if there is one, producing one),

(3) whether at most one automaton exists satisfying a given condition,

(4) whether a given sentence is true.

These results make use of a theorem (5.3) which characterizes finite automata behavior by means of formulas of an arithmetic. The following corollary is typical of the side results obtained. If a natural number is identified with the set of natural numbers less than it, then the first-order theory of quasi-finite (finite or finite complement) sets of natural numbers based upon the Boolean set operations and the property of being a natural number is decidable.

For certain other arithmetics, it is shown that algorithms of the type indicated above fail to exist.

[This paper was printed and distributed as Technical Report 2755-6-T, June, 1959. It has been submitted for journal publication.]

Weak Second-Order Arithmetic and Finite Automata

(Technical Report 2794-6-T, Sept. 1959)

J. Richard Büchi

ABSTRACT

In essence, this paper states that a certain formal language, a weak second-order arithmetic (designated throughout this paper as W.2.A., and in Elgot, Decision Problems of Finite Automata Design and Related Arithmetics, as L_1^1) can be used in place of the formalism of regular expressions (developed by S. C. Kleene in his paper "Representation of Events in Nerve Nets and Finite Automata," in Automata Studies, Princeton University Press, 1956) in denoting the behavior of finite automata. The important Kleene Synthesis and Analysis theorems can also be obtained in this new formulation. (Synthesis: For every formula of W.2.A. one can construct an automaton with special output, such that the behavior of the automaton is just the set of predicates which satisfy the given formulas. Analysis: For every automaton with special output one can obtain a formulas of W.2.A. such that the formula denotes the behavior of the automaton). This result is of particular value because formulas of W.2.A. seem to be more convenient than regular expressions for formalizing conditions on the behavior of automata. Important additional results in pure logic are also obtained: the synthesis and analysis theorems yield valuable information on the strength of W.2.A. and related formalisms. [For expositions of Kleene's theory of regularity see Copi, Elgot, and Wright, "Realization of Events by Logical Nets," JACM, 5, 181-196 (1958; Rabin and Scott, "Finite Automata and Their Decision Problems," IBM Journal, 114-125 (April, 1959); and Myhill, "Finite Automata and Representation of Events," WADC Report TR 57-624, Fundamental Concepts in the Theory of Systems, October, 1957, pp. 112-137.]

[This paper was printed and distributed as Technical Report 2794-6-T, September 1959. It is to be published in 1960 in the Zeitschrift für Mathematische Logik and Grundlagen der Mathematik.]

On a Hierarchy of Monadic Predicate Quantifiers

J. Richard Büchi

ABSTRACT

Let x, x_1, x_2,... be variables ranging over natural numbers, and let i_1, i_2, i_3,... be variables ranging over monadic predicate (sets) of natural numbers, let $\underline{i}$, $\underline{j}$ denote k-tuples of i's. The hierarchy $[\Sigma_m, \Pi_m]$ of predicates $P(\underline{i})$ on predicates is defined thus: Σ_0 = all predicates $P(\underline{i})$ definable by formulas $K[\underline{i}(o)] \vee (\exists x)\, H[\underline{i}(x), \underline{i}(s+1)] \vee (\forall x)\, U[\underline{i}(x)]$, whereby K,H,U are truth functions in the indicated constituents, $\Pi_m = \{\sim P \mid P \in \Sigma_m\}$, $\Sigma_{m+1} = \{(\exists \underline{j})\, P \mid P \in \Pi_m\}$. A k-tuple $\underline{i}$ of monadic predicates may be considered to be an infinite sequence $\underline{i}(0)\ \underline{i}(1)\ \underline{i}(2)$,... whose elements are drawn from the finite alphabet consisting of all k-tuples of truth-values. Let $\underline{u}, \underline{v}$,... denote finite words on this alphabet, let $\underline{u}\ \underline{v}$ denote concatenation. Def: $P(\underline{i})$ is of finite rank if there is a -congruence relation $\underline{u} \sim \underline{v}$ on words, with finite partition, and such that $\underline{u}_q \sim \underline{v}_q$ implies $P(\underline{u}_1 \underline{u}_2 \ldots) \equiv P(\underline{v}_1 \underline{v}_2 \ldots)$. Theorem: The following are equivalent conditions on $P(\underline{i})$: (1) $P \in \Sigma_2$, (2) P is of finite rank, (3) $P = P_1 \cup \ldots \cup P_a$ whereby each P_c is of the form $R S^* S^* \ldots$, R and S being regular sets of finite words. This is established by using basic facts from automata-theory (regularity), the fan theorem (König's infinity lemma), and Ramsey's theorem [Proc. London Math. Soc., (2), 30, 264-286 (1930)]. Clearly P is of finite rank if and only if $\sim P$ is; therefore Corollary: $\Sigma_2 = \Pi_2$ and therefore $\Sigma_n = \Pi_n = \Sigma_2$ for $n \geq 2$.

[This paper was prepared for the Missoula, Montana, meeting of the American Mathematical Society in June, 1960. The abstract was published in the Notices of the American Mathematical Society, Vol. 7, No. 3, issue 46, June, 1960, p.381.]

On a Problem of Tarski

J. Richard Büchi

ABSTRACT

Let SC be the interpreted system containing variables x ranging over natural numbers, variables i ranging over monadic predicates on natural numbers, the successor function, propositional connectives, and quantifiers for both types of variables. Using the theorem and definition of the preceeding abstract one obtains: Theorem 1: The predicates $P(i_1,---,i_k)$ definable in SC are exactly those belonging to Σ_2. For example: exactly the ultimately periodic sets of natural numbers are definable in SC. Theorem 2: Truth of sentences in SC is decidable. This answers a question of Tarski [see R. M. Robinson, Proc. Am. Math. Soc., 9, 238-242 (1958)], and seems to be a rather strong result since essential assertions about infinity can be stated in SC(parts of the fan-theorem and Ramsey's theorem). By interpreting predicates i as binary expansions of real numbers, theorem 2 may be stated thus: Theorem 2': The first-order theory of [Re, +, Nn, Pw] is decidable. Here Re is the set of real numbers ≥ 0, Nn is the set of natural numbers and Pw is the set of integral powers of 2. (Compare this result with Tarski's decidability of >Re, +, .]). Let SC_{per} be like SC, except that the variables i range over ultimately periodic predicates. Theorem 3: A sentence is true in SC_{per} if and only if it is true in SC. i.e., [Re, +, Nn, Pw] and [R, +, Nn, Pw] are arithmetically equivalent. Here R stands for the set of rational numbers.

[This paper was prepared for the Missoula, Montana, meeting of the American Mathematical Society in June, 1960. The abstract was published in the Notices of the American Mathematical Society, Vol. 17, No. 3, June, 1960, p. 382.]

On a problem of Tarski[1]

by J. Richard Büchi

Let SC (sequential calculus) be the interpreted formalism which makes use of individual variables t, x, y, z, ... ranging over natural numbers, monadic predicate variables q( ), r( ), s( ), i( ), j( ), ... ranging over arbitrary sets of natural numbers, the individual symbol o for zero, the function symbol ' denoting the successor function, propositional connectives, and quantifiers for both types of variables.

The purpose of this note is to outline an effective method for deciding truth of sentences in SC. This, according to R. M. Robinson [10], answers a question of Tarski's. In addition, a rather complete understanding of definability in SC will be obtained.

1. Notations. $\underset{\sim}{i}$ denotes a n-tuple of predicate variables. Expressions like $A[\underset{\sim}{i}(o)]$, $B[\underset{\sim}{i}(t), \underset{\sim}{i}(t')]$ denote propositional formulas in the indicated constituents. Σ_n, π_n, denote the classes of formulas of SC of the following type,

Σ_1 : $(\exists \underset{\sim}{r}) \cdot A[\underset{\sim}{r}(o)] \wedge (\forall t)\, B[\underset{\sim}{i}(t), \underset{\sim}{r}(t), \underset{\sim}{r}(t')] \wedge (\exists t)\, C[\underset{\sim}{r}(t)]$

π_1 : $(\forall \underset{\sim}{r}) \cdot A[\underset{\sim}{r}(o)] \vee (\exists t)\, B[\underset{\sim}{i}(t), \underset{\sim}{r}(t), \underset{\sim}{r}(t')] \vee (\forall t)\, C[\underset{\sim}{r}(t)]$

Σ_{n+1} : $(\exists \underset{\sim}{r}) \cdot \underline{F}(\underset{\sim}{i}, \underset{\sim}{r})$, whereby $\underline{F} \in \pi_n$

π_{n+1} : $(\forall \underset{\sim}{r}) \cdot \underline{F}(\underset{\sim}{i}, \underset{\sim}{r})$, whereby $\underline{F} \in \Sigma_n$

The quantifiers $(\exists t)_x^y \underline{A}(t)$ for $(\exists t)\, [x \leqq t < y \wedge \underline{A}(t)]$, $(\forall t)_x^y A(t)$ for $(\forall t)[x \leqq t < y \supset \underline{A}(t)]$, $(\exists^{\omega} t)\, \underline{A}(t)$ for $(\forall x)(\exists t)\, [x < t \wedge \underline{A}(t)]$, $(\forall_{\omega} t)\, \underline{A}(t)$ for $(\exists x)(\forall t)[x < t]\ \underline{A}(t)$, $(\exists j)_{\omega}\, A(j)$ for $(\exists j)\, [(\exists^{\omega} t) j(t) \wedge \underline{A}(j)]$ can be defined in SC. The classes Σ_1^{ω} and π_1^{ω} of formulas are defined as follows,

Σ_1^{ω} : $(\exists r) \cdot A[\underset{\sim}{r}(o)] \wedge (\forall t) B[\underset{\sim}{i}(t), \underset{\sim}{r}(t), \underset{\sim}{r}(t')] \wedge (\exists^{\omega} t)\, c[\underset{\sim}{r}(t)]$

π_1^{ω} : $(\forall r) \cdot A[\underset{\sim}{r}(o)] \vee (\exists t)\, B[\underset{\sim}{i}(t), \underset{\sim}{r}(t), \underset{\sim}{r}(t')] \vee (\forall_{\omega} t)\, C[\underset{\sim}{r}(t)]$

1. The author is much indebted to Dr. J. B. Wright. The work was done under a grant from the National Science Foundation to the Logic of Computers Group, and with additional assistance through contracts with the Office of Naval Research, Office of Ordnance Research, and the Army Signal Corps.

Let $\underset{\sim}{i}$ be a k-tuple of predicates. The 2^k states of $\underset{\sim}{i}$ are the k-tuples of truth-values. $\underset{\sim}{i}$ may be viewed as an infinite sequence $\underset{\sim}{i}(0)\ \underset{\sim}{i}(1)\ \underset{\sim}{i}(2)\ \ldots$ of states. The variables $\underline{u}$, $\underline{v}$, $\underline{w}$, ... will be used for words (i.e., finite sequences) of states. $\underline{u}\frown\underline{v}$ denotes the result of juxtaposing $\underline{u}$ and $\underline{v}$. A congruence-relation is an equivalence relation $\underline{u} \backsim \underline{v}$ on words such that $\underline{u} \backsim \underline{v}$ implies $\underline{u}\frown\underline{w} \backsim \underline{v}\frown\underline{w}$ and $\underline{w}\frown\underline{u} \backsim \underline{w}\frown\underline{v}$. An equivalence relation is of finite rank if it partitions the words into finitely many classes.

Also the following classes of formulas will play an essential role,

$$\Sigma_{reg} : (\exists \underset{\sim}{r}) \cdot A[\underset{\sim}{r}(x)] \wedge (\forall t^{y}_{x}\, B[\underset{\sim}{i}(t), \underset{\sim}{r}(t), \underset{\sim}{r}(t')] \wedge C[\underset{\sim}{r}(y)]$$

$$\pi_{reg} : (\forall r) \cdot A[\underset{\sim}{r}(x)] \vee (\exists t^{y}_{x}\, B[\underset{\sim}{i}(t), \underset{\sim}{r}(t), \underset{\sim}{r}(t')] \vee C[\underset{\sim}{r}(y)].$$

These may be called regular formulas. Note that for a regular formula, $\underline{R}(\underset{\sim}{i},x,y)$ depends only on the word $\underset{\sim}{i}(x)\ \underset{\sim}{i}(x+1)\ \ldots\ \underset{\sim}{i}(y-1)$. If $\underline{\underline{R}}$ is the set of all words $\underset{\sim}{i}(0)\ \underset{\sim}{i}(1)\ \ldots\ \underset{\sim}{i}(h)$ such that $\underline{R}(\underset{\sim}{i},0,h+1)$, then the formula $\underline{R}(\underset{\sim}{i},x,y)$ is said to determine the set of words $\underline{\underline{R}}$.

2. A fundamental lemma on infinite sequences. The working of the decision-method for SC is based on induction and a rather more sophisticated property of infinity, namely Theorem A of Ramsey [9].[2] Essential parts of this theorem can actually be formulated in SC, in the form of a surprising assertion about the division of infinite sequences into consecutive finite parts.

Lemma 1. Let $\underset{\sim}{i}$ be any k-tuple of predicates, and let $\underline{\underline{E}}_0, \ldots, \underline{\underline{E}}_n$ be a partition of all words on states of $\underset{\sim}{i}$ into finitely many classes. Then there exists a division $\underset{\sim}{i}(0)\ \underset{\sim}{i}(1)\ \ldots\ \underset{\sim}{i}(x_1-1),\ \underset{\sim}{i}(x_1)\ \underset{\sim}{i}(x_1+1)\ \ldots\ \underset{\sim}{i}(x_2-1),\ \underset{\sim}{i}(x_2)\ \underset{\sim}{i}(x_2+1)\ \ldots\ \underset{\sim}{i}(x_3-1),\ \ldots$ of $\underset{\sim}{i}$ such that all words $\underset{\sim}{i}(x_p)\ \underset{\sim}{i}(x_p+1)\ \ldots\ \underset{\sim}{i}(x_q-1)$ belong to one and the same of the classes $\underline{\underline{E}}_0, \ldots, \underline{\underline{E}}_n$.

Proof: Assume $i, \underline{\underline{E}}_0, \ldots, \underline{\underline{E}}_n$ are as supposed in lemma 1. For $0 \leqq c \leqq n$ let P_c consist of all $\{y_1,y_2\}$ such that $y_1 < y_2$ and $\underset{\sim}{i}(y_1)\ \underset{\sim}{i}(y_1+1)\ \ldots\ \underset{\sim}{i}(y_2-1) \in \underline{\underline{E}}_c$. Then $P_0, \ldots, P_n$ clearly is a partition of all 2-element sets of natural numbers. By Ramsey's theorem A it follows that there is an infinite sequence $x_1 < x_2 < x_3 < \ldots$ and a $0 \leqq c \leqq n$, such that $\{x_p,x_q\} \in P_c$ for all $x_p < x_q$. By definition of P_c this yields the conclusion of lemma 1.

3. Automata-theory. The following concepts and results are borrowed from the theory of finite automata, and play a very essential role in the study of SC.

---

2. The usefulness of the "Unendlichkeitslemma" of König [5] (also known as "fan-theorem" in its intuitionistic version) in related problems of automata-theory was first observed by Dr. J. B. Wright. Because of its affinity to König's lemma the present application of Ramsey's theorem was suggested.

The reader is referred to Büchi [1], where some of the details are carried out in similar form, and where further references to the mathematical literature on automata are given

Lemma 2. The following are equivalent conditions on a set $\underline{\underline{R}}$ of words:

1) $\underline{\underline{R}}$ is regular.

2) $\underline{\underline{R}} = \underline{\underline{E}}_1 \cup \cdots \cup \underline{\underline{E}}_n$, whereby $\underline{\underline{E}}_1, \ldots, \underline{\underline{E}}_n$ are some of the congruence classes modulo a congruence relation of finite rank on words.

3) $\underline{\underline{R}}$ is the set of words determined by a formula $\underline{F}(\underset{\sim}{i},x,y)$ belonging to Σ_{reg}.

4) $\underline{\underline{R}}$ is the set of words determined by a formula $\underline{G}(\underset{\sim}{i},x,y)$ belonging to π_{reg}.

5) There is an "automata-recursion" $\underset{\sim}{r}(o) \equiv I$, $\underset{\sim}{r}(t') \equiv J[\underset{\sim}{i}(t), \underset{\sim}{r}(t)]$ and an "output" $U[\underset{\sim}{r}(t)]$ such that a word $\underset{\sim}{i}(o)\ \underset{\sim}{i}(1) \ldots \underset{\sim}{i}(x\text{-}1)$ belongs to $\underline{\underline{R}}$ just in case the recursion produces an $\underset{\sim}{r}(x)$ such that $U[r(x)]$ holds.

The reader who is not familiar with the concept of regularity of Kleene [4] may take $1 \leftrightarrow 2$ of the lemma as its definition (note the analogy to ultimately periodic sets of natural numbers). $3 \leftrightarrow 5$ is shown in essence by Myhill's "subset-construction" [6]; nearly in the present form the details are carried out in Büchi [1], lemma 7. By dualization one gets $4 \rightarrow 5$. The assertions $5 \rightarrow 3$. and $5 \rightarrow 4$ are trivial. A proof of $2 \leftrightarrow 3$ is contained essentially in Rabin and Scott [8].

Lemma 3. If the formulas $\underline{R}(\underset{\sim}{i},x,y)$, $\underline{S}(\underset{\sim}{i},x,y)$ determine regular sets of words, then so do the formulas $(\exists t)^{\underset{\sim}{y}}_{x}\ \underline{R}(\underset{\sim}{i},t,y)$, $(\forall t)^{\underset{\sim}{y}}_{x}\ \underline{R}(\underset{\sim}{i},t,y)$, $\underline{R}(\underset{\sim}{i},x,y)\ \ \underline{S}(\underset{\sim}{i},x,y)$, $\underline{R}(\underset{\sim}{i},x,y) \vee \underline{S}(\underset{\sim}{i},x,y)$, and $\sim\underline{R}(\underset{\sim}{i},x,y)$.

This follows by lemma 2 and Skolem's method of replacing bounded quantifiers by recursions.

Lemma 4. If the formula $\underline{R}(\underset{\sim}{i},x,y)$ determines a regular set, then one can find formulas $\underline{D}(\underset{\sim}{i})$ and $\underline{E}(\underset{\sim}{i})$ in Σ_1 (in π_1) such that $\underline{D}(\underset{\sim}{i}) \equiv (\forall t)\ \underline{R}(\underset{\sim}{i},o,t)$ and $\underline{E}(\underset{\sim}{i}) \equiv (\exists t)\ \underline{R}(\underset{\sim}{i},o,t)$.

This follows by $1 \leftrightarrow 5$ of lemma 2. The following lemma will provide the basis for the decision method of SC. It was suggested by, and its proof is typical for, automata theory.

Lemma 5. There is an effective method for deciding truth of sentences $\underline{A}$ in Σ_1^{ω}.

Proof: Let $C(\underset{\sim}{r})$ be a formula of form $K[\underset{\sim}{r}(o)] \wedge (\forall t)\ H[\underset{\sim}{r}(t), \underset{\sim}{r}(t')] \wedge (\exists^{\omega} t)\ L[\underset{\sim}{r}(t)]$. Suppose $\underset{\sim}{r}$ is a k-tuple of predicates such that $\underline{C}(\underset{\sim}{r})$ holds. Then there are $x_1 < x_2 < \ldots$ such that $L[\underset{\sim}{r}(x_1)]$, $L[\underset{\sim}{r}(x_2)]$, ... Because $\underset{\sim}{r}$ has but a finite number of states, there must be a repetition $\underset{\sim}{r}(x_p) = \underset{\sim}{r}(x_q)$ of some state U.

Therefore, $(\exists \underset{\sim}{r})\ \underline{C}(\underset{\sim}{r})$ implies the assertion,

(1) There are words $\underline{x} = X_0X_1 \ldots X_a$ and $\underline{y} = Y_1Y_2 \ldots Y_b$ of states and a state U such that $L[U]$, and $K[X_0] \wedge H[X_0,X_1] \wedge \ldots \wedge H[X_{a-1},X_a] \wedge H[X_a,U]$, and $H[U,Y_1] \wedge H[Y_1,Y_2] \wedge \ldots \wedge H[Y_{b-1},Y_b] \wedge H[Y_b,U]$.

Conversely (1) implies $(\exists \underset{\sim}{r})\ \underline{C}(\underset{\sim}{r})$, because one has but to let $r = \underline{x}\underline{U}\underline{y}\underline{U}\underline{y}\underline{U}\underline{y} \ldots$. Thus, a method I which decides, for given propositional formulas K,H,L and given state U, whether or not (1) holds, will also be a method for deciding truth of Σ_1^ω-sentences $(\exists r)\ \underline{C}(r)$. Clearly such a method I can be composed from a method II which, for given propositional formula H[X,y] and given states V and W, decides whether or not,

(2) There is a word $\underline{x} = X_1X_2 \ldots X_a$ such that $H[V,X_1] \wedge H[X_1,X_2] \wedge \ldots \wedge H[X_{a-1},X_a] \wedge H[X_a,W]$.

Let $n = 2^k$ be the number of states, and note that in a word $\underline{x} = X_1X_2 \ldots X_a$ of length $a > n$ there must occur a repetition $X_p = X_q$, $p < q < a$. Clearly if $\underline{x}$ satisfies (2), then so does the shorter word $\underline{y} = X_1X_2 \ldots X_pX_{q+1}X_{q+2} \ldots X_a$ Therefore, to establish whether or not (2) holds it suffices to check among the finitely many words $\underline{x}$ of length $\leqq n$. This remark clearly yields a method II for (2), whereby lemma 5 is established.

4. Reduction of formulas of SC. The following lemma is obtained by methods similar to those in Büchi [1], lemma 1.

Lemma 6. To every formula $\underline{A}(\underset{\sim}{i})$ of SC one can obtain an equivalent formula $\underline{B}(\underset{\sim}{i})$ belonging to some Σ_n (to some π_n).

Based on lemmas 1 to 4 one now can prove the fundamental fact on reduction of formulas in SC.

Lemma 7. To every formula $A(\underset{\sim}{i})$ in Σ_1^ω (in π_1^ω) one can obtain an equivalent formula $\underline{B}(\underset{\sim}{i})$ in π_1^ω (in Σ_1^ω).

Proof: Suppose $\underline{A}(i)$ is in Σ_1^ω, say

(1) $\underline{A}(\underset{\sim}{i})$: $(\exists r) \cdot K[\underset{\sim}{r}(o)] \wedge (\forall t)\ H[\underset{\sim}{i}(t), \underset{\sim}{r}(t), \underset{\sim}{r}(t')] \wedge (\exists^\omega t)\ L[\underset{\sim}{r}(t)]$.

If V,W are states of $\underset{\sim}{r}$ and $\underline{x} = X_0X_1 \ldots X_h$ is a word of states of $\underset{\sim}{i}$ then define,

$$[V,\underline{x},W]_1 \ : \ \bigvee_{U_1 \ldots U_h} \cdot H[X_0,V,U_1] \wedge H[X_1,U_1,U_2] \wedge H[X_2,U_2,U_3] \wedge \ldots \wedge H[X_h,U_h,W]$$

$$[V,\underline{x},W]_2 \ : \ \bigvee_{U_1 \ldots U_h} \cdot H[X_0,V,U] \wedge \ldots \wedge H[X_h,U_h,W] \wedge [L[U_1] \vee \ldots \vee L[U_h]].$$

(Read $[\ ]_1$ as "there is an H-transition through L vrom V by $\underset{\sim}{x}$ to W.") Next define the binary relation $\backsim$ on words of states of $\underset{\sim}{i}$:

$$x \backsim \underset{\sim}{y} \ : \ \bigwedge_{VW}([V,\underset{\sim}{x},W]_1 \equiv [V,\underset{\sim}{y},W]_1 \quad \bigwedge_{VW}([V,\underset{\sim}{x},W]_2 \equiv [V,\underset{\sim}{y},W]_2)$$

If m is the number of states of $\underset{\sim}{r}$, then clearly $\backsim$ is the intersection of $m^2 + m^2$ dichotomies. Therefore, (2) $\backsim$ is an equivalence relation of finite rank $a \leq 2^{2m^2}$. Furthermore, using the definitions of $[\ ]_1$ and $[\ ]_2$ one obtains,

(3) $\backsim$ is a congruence relation on words. By (2), (3), and lemma 2 it follows that one can find formulas $\underset{\sim}{E}_1(\underset{\sim}{i},x,y)$, ..., $\underset{\sim}{E}_a(\underset{\sim}{i},x,y)$ such that

(4) $\underset{\sim}{E}_1$, ..., $\underset{\sim}{E}_a$ are regular formulas (i.e., belong to Σ_{reg}).

(5) $\underset{\sim}{E}_1$, ..., $\underset{\sim}{E}_a$ determine the congruence classes of $\backsim$.

Next one applies lemma 1 to the partition $\underset{\sim}{E}_1$, ..., $\underset{\sim}{E}_a$. If follows that for any $\underset{\sim}{i}$,

$$(6)\ (\exists s)_\omega(\forall y)(\forall x)_o^y[s(x)s(6)\supset \underset{\sim}{E}_1(\underset{\sim}{i},x,y)] \vee \ldots \vee (\exists s)_\omega(\forall y)(\forall x)_o^y[s(x)s(y)\supset \underset{\sim}{E}_a(\underset{\sim}{i},x,y)].$$

If one defines for $1 \leq c,d \leq a$,

$$\underset{\sim}{F}_{c,d}(i) \ : \ (\exists s)_\omega \cdot (\exists x)[s(x) \wedge \underset{\sim}{E}_c(\underset{\sim}{i},o,x)] \wedge (\forall y)(\forall x)_o^y\ [s(x)s(y) \supset \underset{\sim}{E}_d(\underset{\sim}{i},x,y)]$$

then clearly each disjunct of (6) is equivalent to a disjunction of $\underset{\sim}{F}_{c,d}$'s. Therefore,

$$(7)\ \bigvee_{1 \leq c,d \leq a} \underset{\sim}{F}_{c,d}(\underset{\sim}{i}) \ , \quad \text{holds for all } \underset{\sim}{i}.$$

Suppose now that $\underset{\sim}{F}_{c,d}(\underset{\sim}{i})$ and $\underset{\sim}{F}_{c,d}(\underset{\sim}{j})$. Then, by definition of $F_{c,d}$ and by (5) there are $x_1 < x_2 < x_3 < \ldots$ and $y_1 < y_2 < y_3 < \ldots$ such that

$$\underset{\sim}{i}(o) \ldots \underset{\sim}{i}(x_1\text{-}1) \backsim \underset{\sim}{j}(o) \ldots \underset{\sim}{j}(y_1\text{-}1)$$

$$\underset{\sim}{i}(x_p) \ldots \underset{\sim}{i}(x_{p+1}\text{-}1) \backsim \underset{\sim}{j}(y_p \ldots \underset{\sim}{j}(y_{p+1}\text{-}1) \qquad , \ p = 1,2,3,\ldots$$

Because of the definition of $\backsim$ and (1) it therefore follows that $A(\underset{\sim}{i}) = A(\underset{\sim}{j})$. Thus, if $\underset{\sim}{F}_{c,d}(\underset{\sim}{i}) \wedge \underset{\sim}{F}_{c,d}(\underset{\sim}{j})$ then $\underset{\sim}{A}(\underset{\sim}{i}) \equiv \underset{\sim}{A}(\underset{\sim}{j})$. Or restating this result,

$$(8)\ (\forall i)[\underset{\sim}{F}_{c,d}(\underset{\sim}{i}) \supset \underset{\sim}{A}(\underset{\sim}{i})] \vee (\forall \underset{\sim}{i})[\underset{\sim}{F}_{c,d}(\underset{\sim}{i}) \supset \sim A(\underset{\sim}{i})], \text{ for any } 1 \leq c,d \leq a.$$

If one now defines the set Φ of pairs $1 \leq c,d \leq a$ by,

$$(9)\ \Phi(c,d) \equiv \sim (\exists \underset{\sim}{j})[A(\underset{\sim}{j}) \wedge \underset{\sim}{F}_{c,d}(\underset{\sim}{j})], \text{ for } 1 \leq c,d \leq a$$

it follows by (7) and (8) that

(10) $\sim\underset{\sim}{A}(\underset{\sim}{i}) \equiv \bigvee_{\Phi(c,d)} \underset{\sim}{F}_{c,d}(i)$.

By (4), lemma 3, and lemma 4 it follows that there are formulas $\underset{\sim}{D}_c(\underset{\sim}{i},s)$ and $\underset{\sim}{G}_d(\underset{\sim}{i},s)$ in Σ_1 which are equivalent respectively to $(\exists x)[s(x) \wedge \underset{\sim}{E}_c(\underset{\sim}{i},o,x)]$ and $(\forall y)(\forall x)^y_o[s(x)s(y) \supset \underset{\sim}{E}_d(\underset{\sim}{i},x,y)]$. Referring to the definition of $\underset{\sim}{F}_{c,d}$ this yields,

$$\underset{\sim}{F}_{c,d}(\underset{\sim}{i}) \equiv (\exists s) \cdot (\exists^{\omega} t)\ s(t) \wedge \underset{\sim}{D}_c(\underset{\sim}{i},s) \wedge \underset{\sim}{G}_d(\underset{\sim}{i},s)$$

Because $\underset{\sim}{D}_c$ and $\underset{\sim}{G}_d$ are in Σ_1 it follows,

(11) $\underset{\sim}{F}_{c,d}(\underset{\sim}{i}) \equiv (\exists s\underset{\sim}{p}\underset{\sim}{q}).\ I(o) \wedge (\forall t)\ J(t) \wedge (\exists t)\ M(t) \wedge (\exists t)\ N(t) \wedge (\exists^{\omega} t)\ s(t)$,

for some matrices $I[\underset{\sim}{p}(o),\underset{\sim}{q}(o)]$, $J[\underset{\sim}{i}(t), s(t), \underset{\sim}{q}(t), \underset{\sim}{p}(t), \underset{\sim}{q}(t'), \underset{\sim}{p}(t')]$, $M[\underset{\sim}{q}(t)]$, $N[\underset{\sim}{p}(t)]$. Note that $(\exists t)M(t) \wedge (\exists t)N(t) \wedge (\exists^{\omega} t)s(t)$ may be replaced by $[(\exists^{\omega} x)[(\exists t)^x_o M(t) \wedge (\exists t)^x_o N(t) \wedge s(x)]$, and this in turn may be replaced by $(\exists jh)[\sim j(o) \wedge \sim h(o) \wedge (\forall t)[j(t') \equiv j(t) \vee M(t)] \wedge (\forall t)[h(t') \equiv h(t) \vee N(t)] \wedge (\exists^{\omega} t)[j(t) \wedge h(t) \wedge s(t)]]$. The corresponding substitution in (11) then shows that, for any $1 \leq c,d \leq a$, the formula $\underset{\sim}{F}_{c,d}(\underset{\sim}{i})$ is equivalent to a formula $\underset{\sim}{J}_{c,d}(\underset{\sim}{i})$ in Σ_1^{ω}. By (10) it follows that $\sim\underset{\sim}{A}(\underset{\sim}{i})$ is equivalent to a formula in Σ_1^{ω}, and therefore $\underset{\sim}{A}(\underset{\sim}{i})$ is equivalent to a formula $\underset{\sim}{B}(\underset{\sim}{i})$ in π_1^{ω}. This ends the proof of lemma 7.

Theorem 1. The hierarchy of relations on predicates definable by formulas of Σ_n, π_n collapses at n=2. To every formula $\underset{\sim}{A}(\underset{\sim}{i})$ of SC one can find a formula $\underset{\sim}{B}(\underset{\sim}{i})$ of Σ_1^{ω} (of π_1^{ω}, Σ_2, π_2) which is equivalent to $\underset{\sim}{A}(\underset{\sim}{i})$.

Proof: Let $\underset{\sim}{C}(\underset{\sim}{i})$: $(\exists r)[K(o) \wedge (\forall t)\ H(t) \wedge (\exists t)\ L(t)]$ be any formula of Σ_1. Then $\underset{\sim}{C}(\underset{\sim}{i}) \equiv (\exists \underset{\sim}{r})[K(o) \wedge (\forall t)\ H(t) \wedge (\exists^{\omega} x)(\exists t)^x_o\ L(t)]$. If the term $(\exists t)^x_o\ L(t)$ is dealt with as shown at the end of the proof of lemma 7, this yields a formula $\underset{\sim}{D}(\underset{\sim}{i})$ in Σ_1^{ω} which is equivalent to $C(\underset{\sim}{i})$. Using this remark part one of theorem 1 easily follows by an induction of n and the use of lemma 7. By lemma 6 the other part of theorem 1 follows.

Remarks: 1. The set U consisting of all infinite i can be defined by a Σ_1^{ω} (a Σ_2) formula, but not by a π_1^{ω} (a π_2) formula. This shows that theorem 1 cannot be much improved.

2. Using theorem 1 one easily shows that also formulas $\underset{\sim}{A}(\underset{\sim}{i},x_1, \ldots, x_n)$ of SC, containing individual variables can be put into a normal form, namely

$$(\exists \underset{\sim}{r}) \cdot K[\underset{\sim}{r}(o)] \wedge (\forall t)\ H[\underset{\sim}{i}(t), \underset{\sim}{r}(t), \underset{\sim}{r}(t')] \wedge (\exists^{\omega} t)\ L[\underset{\sim}{r}(t)] \wedge U[\underset{\sim}{r}(x_1)] \wedge \ldots \wedge U[\underset{\sim}{r}(x_n)]$$

This yields rather complete information on definability in SC. For example,

3. A conjecture of Robinson [10]: A relation $\underset{=}{R}(x_1, \ldots, x_n)$ on natural numbers is definable in SC if and only if it is definable in SC_{fin}, which is like SC except that the variables i, j, r, ... range over finite sets of natural numbers. This follows from remark 2 by methods similar to those in the proof of lemma 5. For complete discussion of definability in SC_{fin} see Büchi [1].

4. A relation $\underline{\underline{R}}(i_1, \ldots, i_n)$ on finite sets of natural numbers is definable in SC if and only if it is definable in SC_{fin}.

5. Analyzing the proof of lemma 7 one obtains: A set $\underline{\underline{R}}$ of n-tuples $\underset{\sim}{i}$ of predacates is definable in SC if and only if $\underline{\underline{R}} = \underline{\underline{S}}_1 \cup \ldots \cup \underline{\underline{S}}_k$ whereby each $\underline{\underline{S}}_c$ is of the form $\underline{\underline{A}}^\frown \underline{\underline{B}}^\frown \underline{\underline{B}}^\frown \underline{\underline{B}}^\frown \ldots$, $\underline{\underline{A}}$ and $\underline{\underline{B}}$ being regular sets of words.

Lemmas 2, 3, 4, 6 can be stated and proved in a strong constructive version. To see that this also holds for lemma 7, it remains to ascertain that the finite set Φ of pairs, defined by (9) in the proof of 7, may be obtained effectively for a given $\underline{A}(\underset{\sim}{i})$. This follows by lemma 5, if one observes that $\underline{A}(\underset{\sim}{i})$, $\underline{F}_{c,d}(\underset{\sim}{i})$ and therefore $(\exists j)[\underline{A}(j) \wedge \underline{F}_{c,d}(j)]$ are equivalent to Σ_1^ω formulas. It now follows that theorem 1 can also be proved in an effective version. In particular, to every sentence $\underline{A}$ of SC one can effectively construct an equivalent one belonging to Σ_1^ω. Applying lemma 5 again this yields,

Theorem 2. There is an effective method for deciding truth of sentences in SC.

The strength of these results is best seen by noting some very special cases which occur in the literature and have been obtained by rather divergent methods:

1. The decidability of Σ_2-sentences of SC contains the result of Friedman [3], and implies the existence of various other algorithms of finite automata theory as programmed by Church [2]. It also implies some of the results of Wang [11].

2. In SC one can define $x = y$, $x < y$, $x \equiv y \pmod{k}$ (for $k=1,2,\ldots$). The decidability of SC therefore considerably improves a result of Putnam [7].

3. In SC one can define "i is finite." Theorem 2 therefore implies the decidability of SC_{fin}, which was also proved in Büchi [1], and according to Robinson [10] is due to A. Ehrenfeucht.

4. The decidability of the first order theory of [Nn, +, Pw] follows from theorem 4 and improves the classical result of Presburger.

5. Theorem 2 is closely related to another classical result, namely the decidability of the monadic predicate calculus of second order, proved first by Th. Skolem and later by H. Behmann. A modified form of lemma 6 yields a rather simple solution to this problem.

BIBLIOGRAPHY

1. J. R. Büchi, "Weak second order arithmetic and finite automata." Zeitschrift für Math. Log. und Grundl. der Math. (1960), pp.

2. Alonzo Church, "Application of Recursive Arithmetic to the Problem of Circuit Synthesis," Notes of the Summer Institute of Symbolic Logic, Cornell, 1957, pp. 3-50, and "Application of Recursive Arithmetic in the Theory of Computing and Automata," Notes: Advanced Theory of the Logical Design of Digital Computers, U. of Michigan Summer Session, 1959.

3. Joyce Friedman, "Some Results in Church's Restricted Recursive Arithmetic," Journal of Symbolic Logic, 22, pp. 337-342 (1957).

4. S. C. Kleene, "Representation of Events in Nerve Nets and Finite Automata." Automata Studies, Princeton Univ.Press, 1956, pp. 3-41.

5. Denes König, Theorie der endlichen und unendlichen Graphen. Akad. Verlagsges., Leipzig, 1936.

6. John Myhill, "Finite Automata and Representation of Events," WADC Report TR 57-624 Fundamental Concepts in the Theory of Systems, October 1957, pp. 112-137.

7. Hillary Putnam, "Decidability and essential undecidability." Journal of Symbolic Logic, 22 (1957), pp. 39-54.

8. M. Rabin and D. Scott, "Finite Automata and Their Decision Problems." IBM Journal, April, 1959, pp. 114-125.

9. F. P. Ramsey, "On a problem of formal logic." Proc. London Math. Soc. (2) 30 (1929), pp. 264-286.

10. R. M. Robinson, "Restricted set-theoretical definitions in arithmetic." Proc. Am. Math. Soc., 9 (1958), pp. 238-242.

11. Hao Wang, "Circuit synthesis by solving sequential Boolean equations." Zeitsch. für Math. Logik und Grundl. der Math., 5 (1959), pp. 291-322.

SEQUENCE GENERATORS AND DIGITAL COMPUTERS

A. W. Burks
J. B. Wright

The University of Michigan

This research was supported by Project MICHIGAN (Contract No. DA-36-039-sc-52654 of the U. S. Army Signal Corps), the Office of Naval Research (Contract No. Nonr 1224 (21)), and the Office of Ordnance Research (Contract No. DA-20-018-ORD-16971).

This paper grew out of some researches on well-behaved nets (see Sec. 3.1); Hao Wang participated in these early investigations and supplied an essential part of the proof of Lemma 3.3-1 for the case of well-behaved nets.

J. Richard Büchi has made many helpful suggestions during the course of our work.

(An early version of this paper was presented at a seminar session of the International Conference on Information Processing, Paris, June, 1959. An abstract of this early version should appear in the Proceedings of that conference.)

INTRODUCTION

1.1 Sequence Generators

The basic concept of this paper, that of sequence generator, is a generalization of the concepts of digital computer, finite automaton, logical net, and other information processing systems. In this subsection, we will define sequence generator and some related concepts and will illustrate them immediately thereafter.

Definitions: A sequence generator $\Gamma = (S, G, R, P^1, \ldots, P^n)$ consists of a set S (whose elements are called complete states), a set G (whose elements are called generators), a binary relation R (called the direct transition relation), and functions $P^1, \ldots, P^n$ (called projections), for some $n = 0, 1, 2, 3, \ldots$, satisfying the conditions: (1) S is finite, (2) G is a subset of S, (3) R is defined on S_X, and each P^i (for $i = 1, 2, \ldots, n$) is also defined on S. The values of the function P^i, which may be entities of any kind, are called P^i-states.

A sequence generator may be represented by a finite-directed graph whose vertices denote complete states and whose arrows indicate when the direct transition relation holds between two states. In our diagram, we will use rectangles at those vertices which represent generator states and circles at vertices representing complete states which are not also generators; the names of complete states and of P-states are written in the circles and rectangles, see Figures 1.2-1b, 1.2-2b, 1.3-1, etc. Though our diagrams are closely related to the usual state diagrams (transition diagrams) employed to represent automata (see, for example, Moore, 1956, p. 134) there are very significant differences. The vertices (nodes) of our diagrams represent complete states, while in the usual state diagrams the nodes represent internal states. This difference results from the fact that in sequence generators complete states are basic and input and output states are derived from complete states by means of projections, while in the usual approach complete states are derived by compounding internal states and input states. (The latter process is explained in Section 1.2; we will discuss the relation of the two approaches further in Section 2.1.)

Some comments and explanations concerning the definition of sequence generator may be helpful. If n=0 then $\Gamma = (S, G, R)$ is a sequence generator with no projections. Though our definiton of sequence generator permits any number of projections, in this paper we will be mainly interested in sequence generators with zero, one or two projections.

Furthermore, the set of complete states S of a sequence generator may be a null set; in this case the domain of definition of each function P^i will be empty. It is worth noting that essentially (but not quite) the same concept of sequence generator can be obtained without using the set S of complete states in the definition and then defining S to be the union of G and the field of R.

We will use $[\alpha](j,k)$ (where j is a non-negative integer, k is a non-negative integer or $k = \omega$; $j \leq k$) to denote the sequence $\langle\alpha(j), \alpha(j+1),\ldots,\alpha(k)\rangle$ when k is finite and the sequence $\langle\alpha(j), \alpha(j+1), \alpha(j+2),\ldots\rangle$ when $k = \omega$. If P is a projection, $P([\alpha](j,k))$ abbreviates the sequence $\langle P(\alpha(j)), P(\alpha(j+1)),\ldots,P(\alpha(k))\rangle$ when k is finite and the sequence $\langle P(\alpha(j)), P(\alpha(j+1)), P(\alpha(j+2)),\ldots\rangle$ when $k = \omega$.

Definitions: Let $\Gamma = (S,G,R,P^1,\ldots,P^n)$ be a sequence generator and let k be a non-negative integer or ω. $[s](0,k)$ is Γ-sequence if (1) $s(0)\in G$ and (2) for each j, $j<k$, $R(s(j), s(j+1))$. A complete state s is $\left\{\begin{array}{l}\Gamma\text{-accessible}\\ \Gamma\text{-admissible}\end{array}\right.$ [Γ-admissible] if s occurs in some $\left\{\begin{array}{l}\text{-----}\\ \text{infinite}\end{array}\right.$ [infinite] Γ-sequence.

These concepts may be illustrated by reference to the direct transition diagram of Figure 1.3-1a. The sequence $\langle s_7, s_8\rangle$ is a Γ-sequence, while the sequence $\langle s_3, s_4, s_5, s_6, s_4, s_6, s_4, s_6, s_4, s_6, \ldots\rangle$ is an infinite Γ-sequence. Complete states s_7, s_8, and s_9 are Γ-accessible but not Γ-admissible; complete states s_3, s_4, s_5, and s_6 are Γ-accessible and Γ-admissible, while states s_0, s_1, s_2, and s_{10} are inaccessible (and hence inadmissible).

Definitions: Let ρ be a binary relation and α a set; we define

$$\rho(\alpha) = \{ y \mid (\exists x)\rho(x,y) \;\&\; x\in\alpha\}$$

A complete state s of $\Gamma = (S, G, R, P^1,\ldots, P^n)$ is a terminal state of Γ if $R(\{s\})$ is null

A terminal state of Γ is a complete state for which there is no successor by the direct transition relation R. Complete states s_8

In Burks and Wright, 1953, p. 1364 we defined the concept of an admissible state of a net. When a net is converted into a sequence generator (see Sec. 1.2 below) these states will be accessible rather than admissible in the senses of these terms defined above.

and s_{10} are the terminal states of Figure 1.3-1a. Note that if Γ has no terminal states, every Γ-accessible state is Γ-admissible and vice-versa.

We will sometimes need to combine several projections to make a composite projection of them. For this we will use the notation

$$P^1 \times P^2 \times \ldots \times P^n$$

which is defined by

$$[P^1 \times P^2 \times \ldots \times P^n](s) = \langle P^1(s), P^2(s), \ldots, P^n(s) \rangle$$

1.2 Special cases of sequence generators

Many concepts in the theory of information processing turn out to be special cases of the concept of sequence generator or are closely related to this concept. We will discuss a number of these in the present subsection. Since digital computers (automata) and logical nets are of special interest to us we will show in detail how the concept of sequence generator applies to them. In later sections we will derive both new and old results about automata and nets from our new theory of sequence generators.

We will begin with well-formed nets, review the method of deriving a finite automaton from a well-formed net, and then show how to derive a sequence generator from a finite automaton. We will use the definition of well-formed net of Burks and Wright, 1953, p. 1361, modified to allow arbitrary switching elements and delay elements whose initial output states are one as well as delays whose initial output states are zero. In net diagrams certain nodes (junctions) are designated as net outputs and are distinguished by stars. See Figure 1.2-1a.

A well-formed net (w.f.n.) may be analyzed in terms of its input states, delay output states, and net-output states. A digital computer represented by w.f.n. operates as follows. The "state" of a net at a given time is determined by its input state i and its delay-output state d at that time; these pairs $\langle i,d \rangle$ are called the complete states of the net. For each time t (t = 0,1,2,...) the complete state $\langle i,d \rangle$ determines the net output state o at the same time (t) in accordance with an output

---

Sequence generators may also be derived from automata containing delays whose initial output states are unspecified; these are called "abstract delays" in Burks and Wang, 1953, p.201 and Burks, 1957, sec. 3. But we will not complicate the present discussion by considering automata with such delay elements.

function λ, i.e., $\theta = \lambda(i,d)$. At time 0 the delay-output state d_0 is uniquely determined by the initial delay-output states of the delay elements. For each time t the complete state $\langle i,d \rangle$ determines the delay-output state d_1 at the next moment of time (t+1) in accordance with a direct transition function τ, i.e., $d_1 = \tau(i,d)$. The net of Figure 1.2-1a is a well-formed net which represents a binary counter. A is the input node, the starred node C is its net output node, and B its delay output node (the initial state of B is zero). The state of C at t indicates the binary count, i.e., the number modulo 2 of 1's which have appeared (during the interval of time $0,\ldots,t$) on the input node A. The state analysis is given by the following table, where 0 is the initial delay-output state.

i(t)	d(t)	d(t+1) $= \tau(i,d)$	θ(t) $= \lambda(i,d)$
A	B	B	C
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

Definition: A finite automaton is a sextuple $\langle \{i\}, \{d\}, \{\theta\}, d_0, \tau, \lambda \rangle$ where $\{i\}$, $\{d\}$, $\{\theta\}$ are finite non-empty sets (whose elements are called input states, internal states, and output states respectively), $d_0 \in \{d\}$ (d_0 is called the initial internal state), τ is a function from the Cartesian product $\{i\} \times \{d\}$ into $\{d\}$ (called the direct transition function), and λ is a function from the Cartesian product $\{i\} \times \{d\}$ onto $\{\theta\}$ (called the output function). (This is essentially the definition of Burks and Wang, 1956, p. 203; see also Moore, 1956, p. 133.) The procedure for analyzing a well-formed net which is described in the preceding paragraph clearly converts a well-formed net into a finite automaton. This procedure is reversible; that is, given a finite automaton, one can construct a well-formed net which realizes it. Thus the concepts of well-formed net and finite automaton are basically equivalent and either can be taken as a formal definition of the concept "finite digital computer". (See Church, 1955, Kleene, 1956, p. 5, and Burks, 1957, Sec. 3 for other definitions of these concepts.)

A three-projection sequence generator $\Gamma = (S, G, R, I, \Theta, D)$ may be associated with a finite automaton as follows. The elements of S are the complete states $\langle i,d \rangle$ and the elements of G are the complete states $\langle i,d_0 \rangle$. The direct transition relation is defined by $R(\langle i_1,d_1 \rangle, \langle i_2,d_2 \rangle) \equiv [d_2 = \tau(i_1,d_1)]$ and the input, output, and internal state projections by $I(\langle i,d \rangle) = i$, $\Theta(\langle i,d \rangle) = \lambda(i,d)$, and $D(\langle i,d \rangle) = d$ respectively. The sequence generator associated with the binary counter

of Figure 1.2-1a is represented by Figure 1.2-1b. As before, the rectangles represent elements of G. The subscripts on the complete states correspond to the nodes of the counter in the order A, B. $\langle s_{00}, s_{10}, s_{11}, s_{00}, s_{10}, s_{01} \rangle$ is an example of a finite Γ-sequence; in it the input sequence $i_0, i_1, i_1, i_0, i_1, i_0$ produces the output sequence $e_0, e_1, e_0, e_0, e_1, e_1$ (and thus three "ones" on the input leave the counter recording "one"). Note that though the internal states d_0, d_1 are represented on Figure 1.2-1b, the nodes of the graph correspond to complete states and not to internal states as in the case with the usual state diagrams used to represent automata.

We have shown how to transform a well-formed net into a finite automaton and vice-versa. We have also shown how to derive a sequence generator from a finite automaton. The latter process is not in general reversible. Only certain sequence generators (those which are deterministic) may be realized by finite automata (see Section 2.1).

Our next application of sequence generators is to arbitrary "nets", including nets that are not well-formed. We will use the concept of Burks and Wright, 1953, p. 1353, modified to allow arbitrary switching elements and both kinds of concrete delays. Each switch element translates into a switch equivalence which gives the state of the switch output as a truth function of the switch input, and each delay element translates into two delay equivalences, called the "initial delay equivalence" and the "recursive delay equivalence". The initial-delay equivalence gives the initial state of the delay output and the recursive-delay equivalence equates the delay output at any time other than 0 to the delay input at the previous time. Hence, each net translates into a conjunction of equivalences. If the net is not well-formed this conjunction will not directly correspond to (will not give the structure of) a digital computer, but it may specify a computation or behavior condition on a digital computer, (see Section 4) and on this account is of interest. In a sequel to this paper we will show how formulas of a much more general kind can be reduced to conjunctions of net equivalences (see Section 4). Figure 1.2-2a shows a net with input node E and output node F. The non-input switch element driving node A represents the contradictory or "always false" truth function. The initial state of the delay AB is "true", which for coding reasons we represent by "1"; the initial state of the delay FC is 0. The switch equivalences for this net are $A(t) \equiv 0$, $F(t) \equiv F(t)$, and $C(t) \equiv [E(t) \,\&\, \overline{B(t)}]$. $B(0) \equiv 1$ and $C(0) \equiv 0$ are the initial delay equivalences, while $B(t+1) \equiv A(t)$ and $C(t+1) \equiv F(t)$ are the recursive delay equivalences.

The procedure about to be described will, of course, associate a well-formed net with a sequence generator; in fact, this sequence generator will generally be different from either of two sequence generators associated with the well-formed net by the processes described above.

A two-projection sequence generator $\Gamma = (S, G, R, I, \Theta)$ may be associated with an arbitrary net in the following way: A complete state s is an assignment of a truth value to each node of the net which makes the switch equivalences of the net true. An element s of S is a generator (element of G) if s assigns to the delay output nodes truth values which make the initial delay equivalences true. $R(s_1,s_2)$, where $s_1,s_2, \in S$, if and only if the truth values which s_1 assigns to the delay-input nodes and the truth values which s_2 assigns to the delay output nodes satisfy the recursive delay equivalences. For each complete state s, $\{I(s)\}$ $[\Theta(s)]$ is s cut down to the {input} [output] nodes (i.e., $\{I(s)\}$ $[\Theta(s)]$ is the net {input} [output] state contained in s); the input projection will not exist if there are no input nodes.

The sequence generator $\Gamma = (S, G, R, I, \Theta)$ associated with Figure 1.2-2a is represented by Figure 1.2-2b. Though there are six nodes in the net there are only eight complete states. The subscripts on the state symbols s_1, s_{37}, etc. are the decimal codings of the binary representations of the states of the nodes taken in the order E, A, B, C, F, G; e.g., the subscript on s_9 decodes into 001001 showing that in this state nodes B and G are active while the remaining nodes are inactive. The subscript of the input state i is the state of node E and the subscript on the output state θ is the state of node F. $\langle s_{10}, s_{37}, s_2, s_{38}, s_{37}\rangle$ is a Γ-sequence which has a derived input sequence $\langle i_0, i_1, i_0, i_1, i_1\rangle$ and a derived output sequence $\langle \theta_0, \theta_1, \theta_0, \theta_0, \theta_1\rangle$. It can be proved that $F(t) \equiv \sim E(t+1)$; such behavior would not, of course, be possible in a well-formed net.

Our process for associating a sequence generator with an arbitrary net is different from our process for associating a sequence generator with a well-formed net in the following basic respect. In the latter case we first defined input states, delay-output states (internal states), and output states for the net, and then compounded complete states from input states and internal states. On the other hand, in associating a sequence generator with an arbitrary net we first defined states (complete states) over every node, and then derived input and output states by means of projections. It turns out that in general not every assignment of truth values to the input nodes of an arbitrary net is an input state. In fact, we know of no way of defining states for parts of a net(input, internal, and output states) without pre-supposing states for the whole net (complete states). Indeed, it was our work with arbitrary nets which led us to consider sequence generators (in which complete states are basic, input, internal, and output states derivative).

We will now mention other entities besides nets and well-formed nets (digital computers) which are either sequence generators or are closely related to sequence generators. The concept of a non-deterministic automaton of Rabin and Scott (1959, Definition 9) is quite similar to our concept of a sequence generator. Sequence generators are in a certain sense equivalent to formulas constructed from truth functional connectives, monadic predicates, one individual variable "t" (which ranges over discrete times), the successor function, and zero; we plan to develop this connection in a sequel to the present paper (see Section 4). The following are special cases of sequence generators: a finite state grammer (Chomsky and Miller, 1958, p. 95); sequential circuits representable in combinatory logic (Fitch, 1958, p. 263); incompletely specified automata, i.e., automata in which certain sequences of input states are proscribed (Aufenkamp and Hohn, 1957, Section IV); automata with terminal states (ibid., Section VII); and the flow diagrams used in programming a digital computer. A state diagram may be used to characterize a class of finite sequences defined by a regular expression (Myhill, 1957). Finite graphs may be used to analyze certain games (König, 1936; and McKinsey, 1952, Chapter 6). There is an obvious relation between finite graphs and sequence generators, and hence some problems concerning games may be studied by means of sequence generators; we will give an example in the next subsection. We remark finally that Harary and Paper (1957) in applying relational logic to linguistics use ideas closely related to the concept of sequence generator.

Though we have noted a number of applications of the concept of sequence generator, we wish to make it clear that we are not attempting in the present paper to solve all the problems that have been considered for these applications. In the next subsection we will establish some results concerning infinite Γ-sequences for sequence generators without projections. In Section 2 we will treat some concepts in which a single projection plays an essential role, and in Section 3 we will work with concepts in which two projections play a special role.

1.3 Reduced form algorithm

Algorithm plays a fundamental role in this paper, so we will make a few informal comments about them. An algorithm presupposes a well-defined set of entities, called "the domain of the algorithm". An algorithm is a finite system of rules which may be mechanically applied to any entity of its domain. An algorithm which terminates in a finite number of steps when applied to any entity of its domain is called a "terminating algorithm". The Reduced Form Algorithm to be described soon

is a terminating algorithm, since when it is applied to any sequence generator it will eventually terminate in a sequence generator. An algorithm with a domain D is called a decision procedure for a class A which is a subset of D if for every element of D which belongs to A the algorithm terminates in "yes" and for every element of D which does not belong to A the algorithm terminates in "no". The truth table procedure is a decision procedure for the class of tautologies of the propositional calculus.

Before formulating the Reduced Form Algorithm we will describe informally what it does. Let us call a state s of a sequence generator $\Gamma = (S, G, R, P^1, \ldots, P^n)$ "extendable" if there is an infinite sequence of complete states $\langle s(0), s(1), s(2), \ldots \rangle$ such that s (0) is s and R [s(i), s (i+1)] for i = 0, 1, 2, ... (Note that s is not necessarily a generator, and so the infinite sequence of complete states is not necessarily a Γ-sequence.) In Figure 1.3-1 states s_0 and s_1 are extendable, while states s_7 and s_{10} are not. The Reduced Form Algorithm may be applied to any sequence generator Γ. In part 1 of the algorithm the operation of deleting terminal states is iterated until we arrive at a sequence generator $\ddot{\Gamma}$ with no terminal states. Since a sequence generator has non-extendable states if and only if it has terminal states, $\ddot{\Gamma}$ is essentially the result of deleting all nonextendable states from Γ. In part 2 of the algorithm one begins with the generators of $\ddot{\Gamma}$ (and hence of Γ), and by a succession of steps obtains the accessible states of $\ddot{\Gamma}$. A new sequence generator Γ^+, called the reduced form of Γ, is defined on the basis of the states so obtained. Since a state is admissible if and only if it is both extendable and accessible Γ^+ is just Γ cut down to its admissible states (see Theorem 1.3-1).

Algorithm (Reduced Form Algorithm): Consider any sequence generator $\Gamma = (S, G, R, P^1, \ldots, P^n)$.

(1) Form a new sequence generator by deleting all the terminal states of Γ. Iterate this process until you arrive at a sequence generator with no terminal states. Call this final sequence generator $\ddot{\Gamma} = (\ddot{S}, \ddot{G}, \ddot{R}, \ddot{P}^1, \ldots \ddot{P}^n)$.

(2) Define A_1 inductively by

$$A_0 = \ddot{G}$$

$$A_{i+1} = \ddot{R}(A_i)$$

Form the sequence $A_0, A_1, A_2, \ldots$, stopping when $A_{m+1} \subseteq U_{i=0}^m A_i$. (Note: "$\alpha \subseteq \beta$" means that α is either included in β or equals β.) Let $\dot{S} = U_{i=0}^m A_i$, $\dot{G} = \ddot{G}$, and let $\{\dot{R}\}$ $[\dot{P}^i]$ be the {relation R} [projection P^i] cut down to $\dot{S}$. Define $\Gamma^+ = \dot{\Gamma} = (\dot{S}, \dot{G}, \dot{R}, \dot{P}^1, \ldots, \dot{P}^n)$.

We will illustrate the Reduced Form Algorithm. Let $\Gamma = (S,G,R)$ be the sequence generator represented by Figure 1.3-1a, with those complete states which belong to G being designated by rectangles. In part 1 of the algorithm we delete states s_8 and s_{10}, then state s_7, and then state s_9. $\ddot{S}$ consists of the remaining complete states, $\ddot{G}$ contains s_3 and s_6, and $\ddot{R}$ is R cut down to $\ddot{S}$. We have at the end of part 1 the sequence generator $\ddot{\Gamma}$ represented by the result of deleting everything to the right of state 5 in Figure 1.3-1a. In part 2 of the algorithm we form the sequence $\{s_3,s_6\}$ $[=A_0]$, $\{s_4\}$ $[=A_1]$, $\{s_5,s_6\}$ $[=A_2]$, $\{s_4,s_6\}$ $[=A_3]$. Simultaneously we form the sequence $\{s_3,s_6\}$ $[=U_{i=0}^{0}A_i]$, $\{s_3,s_4,s_6\}$ $[=U_{i=0}^{1}A_i]$, $\{s_3,s_4,s_5,s_6\}$ $[=U_{i=0}^{2}A_i]$, stopping at this point since $\{s_4,s_6\} \subset \{s_3,s_4,s_5,s_6\}$, i.e., $A_3 \subset U_{i=0}^{2}A_i$. Hence $S = \{s_3,s_4,s_5,s_6\}$ and Γ^+, the reduced form of Γ, is represented by Figure 1.3-1b.

Theorem 1.3-1: The Reduced Form Algorithm, when applied to any sequence generator Γ, always terminates in a sequence generator Γ^+. The set of complete states of Γ^+ equals the set of Γ-admissible complete states.

As a step toward proving this theorem we first establish Lemma 1.3-2: let ρ be a binary relation and δ_0 a set. Define δ_i for $i = 1,2\ldots$ inductively by $\delta_{i+1} = \rho(\delta_i)$ and let $\alpha_\ell = U_{i=0}^{\ell}\delta_i$ for $\ell = 0,1,2,\ldots$. If, for some j, $\alpha_j = \alpha_{j+1}$ then for all ℓ, $\alpha_\ell \subset \alpha_j$.

Proof: We note first that since the operator ρ may be distributed over the union, $\alpha_{\ell+1} = \alpha_\ell \cup \rho(\alpha_\ell)$. We now assume $\alpha_j = \alpha_{j+1}$ and prove that $\alpha_\ell \subset \alpha_j$, proving first by induction that for all $\ell \geq j$, $\alpha_\ell = \alpha_j$. The initial step is covered by the hypothesis that $\alpha_{j+1} = \alpha_j$. For the general step assume $\alpha_k = \alpha_j$, where $k>j$. By the fact noted above $\alpha_{k+1} = \alpha_k \cup \rho(\alpha_k)$ and $\alpha_{j+1} = \alpha_j \cup \rho(\alpha_j)$. Combining the four preceding equalities we get $\alpha_{k+1} = \alpha_j$. To conclude the proof we note that it follows directly from the definition of α_ℓ that for $\ell<j$, $\alpha_\ell \subset \alpha_j$.

We turn now to the proof of Theorem 1.3-1. We will use freely the notation of the algorithm. (I) We prove first that the Reduced Form Algorithm, when applied to any sequence generator Γ, always terminates in a sequence generator Γ^+. Since S is a finite set, the first part of the algorithm terminates in a sequence generator $\ddot{\Gamma}$. The criterion for stopping in part 2 of the algorithm is based on a monotonically increasing sequence of subjects, of $\ddot{S}$, which is a finite set, so the second part of the algorithm will always terminate. Finally, it is clear that a sequence generator Γ^+ is defined in part 2 of the algorithm. (II) We prove next that the set of complete states of Γ^+ equals the set of Γ-admissible complete states. (IIA) We consider a Γ-admissible complete state s_1 and show that $s_1 \in \ddot{S}$. Since S_1 is Γ-admissible there is an infinite sequence $[s](0,\omega)$ of Γ-admissible states such that for some k, $[s](k) = s_1$. A complete state of Γ is deleted

by part 1 of the algorithm only if it cannot belong to an infinite Γ-sequence, and so $[s](0,k)$ is a $\ddot{\Gamma}$-sequence. Hence by the definition of A_k in the algorithm, $s_1 \in A_k$ and $s_1 \in U_{i=0}^{k} A_i$. We now apply Lemma 1.3-2, letting $\rho = \ddot{R}$ and $\delta_0 = \ddot{G}$. By (I) above, part 2 of the algorithm terminates; in the notation of the algorithm $A_{m+1} \subset U_{i=0}^{m} A_i$. The result of Lemma 1.3-2, put in this notation, is that for all ℓ, $U_{i=0}^{\ell} A_i \subset U_{i=0}^{m} A_i$. We have already shown that $s_1 \in U_{i=0}^{k} A_i$, and so $s_1 \in U_{i=0}^{m} A_i$. But in the algorithm $\dot{S}$ is defined to be $U_{i=0}^{m} A_i$ and so $s_1 \in \dot{S}$. (IIB) We next consider a complete state $s_1 \in \dot{S}$ and show that s_1 is Γ-admissible. In the notation of the algorithm $\dot{S} = U_{i=0}^{m} A_i$ and so $s_1 \in U_{i=0}^{m} A_i$. Hence s_1 is $\ddot{\Gamma}$-accessible. Part 1 of the algorithm terminates in a sequence generator $\ddot{\Gamma}$ with no terminal states. As remarked in Sec. 1.1 every accessible state of a sequence generator with no terminal states is an admissible state. Consequently there exists an infinite $\ddot{\Gamma}$-sequence $[s](0,\omega)$ such that for some k, $s_1 = [s](k)$. By the nature of part 1 of the algorithm $[s]\ (0,\omega)$ is also an infinite Γ-sequence, and so s_1 is Γ-admissible.

Corollary 1.3-3: (a) Every complete state of Γ^+ is Γ^+-admissible. (b) The set of infinite Γ-sequences equals the set of infinite Γ^+-sequences. (c) Every finitie Γ^+-sequence is an initial segment of an infinite Γ-sequence.

We will next discuss the Reduced Form Algorithm and some alternatives to it. Applied to an arbitrary sequence generator Γ part 1 of the Reduced Form Algorithm produces the set of extendable states of Γ. Applied to an arbitrary sequence generator Γ part 2 of the algorithm produces the set of Γ-accessible states. Since a complete state is admissible if and only if it is both extendable and accessible the two parts of the Reduced Form Algorithm applied to a sequence generator Γ in either order produce the same sequence generator Γ^+. A sequence generator Γ derived from a well-formed net in the way indicated in Section 1.2 has no terminal states; consequently when part 2 of the Reduced Form Algorithm is applied to Γ it produces Γ^+.

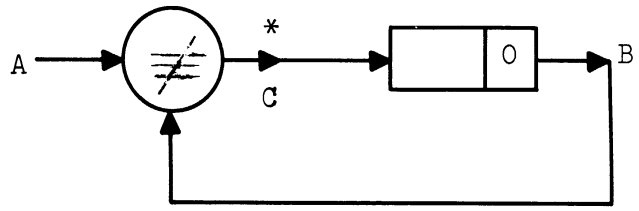
There is an alternative procedure for finding the Γ-admissible complete states of a sequence generator. Let x be the number of complete states of Γ. Form all Γ-sequences of length x+1; it can be proved that a state is Γ-accessible if and only if it occurs in one of these sequences. To find the Γ-admissible states we operate on each sequence as follows: proceeding through the sequence $\langle s(0), s(1),\ldots,s(x)\rangle$ check an occurrence of a state whenever that state has occurred earlier in the same sequence; then delete all states which follow the last checked state. It can be shown that a state is Γ-admissible if and only if it occurs in one of the

resultant sequences. This method of finding the Γ-admissible states can be made the essence of an alternative reduced form algorithm which is simpler to formulate and easier to prove adequate than our Reduced Form Algorithm. It is less efficient however: in the example given earlier m = 2 while x = 11. These differences seem to result from the following fundamental difference between these two algorithms. In the Reduced Form Algorithm the length of the computation is not specified in advance; rather, parts 1 and 2 each contain an internal "stop criterion": one proceeds until he is stopped by these criteria. In contrast, the alternative algorithm first establishes the length of the computation on the basis of a general property of the sequence generator (the number of complete states); since this length is established a priori it is of course determined by the worst case, even though in most cases far fewer steps would have sufficed. This is analogous to the contrast between asynchronous circuits, in which completion of an operation is sensed and the next operation begun immediately, and synchronous circuits, in which the same amount of time is allowed for a given operation in every case, and this is, of course the time required for the worst case (plus a "safety factor"!). We have presented the more efficient of these two algorithms, despite the fact that it is more difficult to formulate and prove adequate. We did this because finding the reduced form is basic to so many automata algorithms; see, for example, Section 2.3 and Section 3.4. But though in many later cases we know of more efficient algorithms (see for example, the alternative to the h-univalence Decision Procedure in Section 3.4), we will not present them because we feel that perspicuity of theory and simplicity of exposition is more important there.

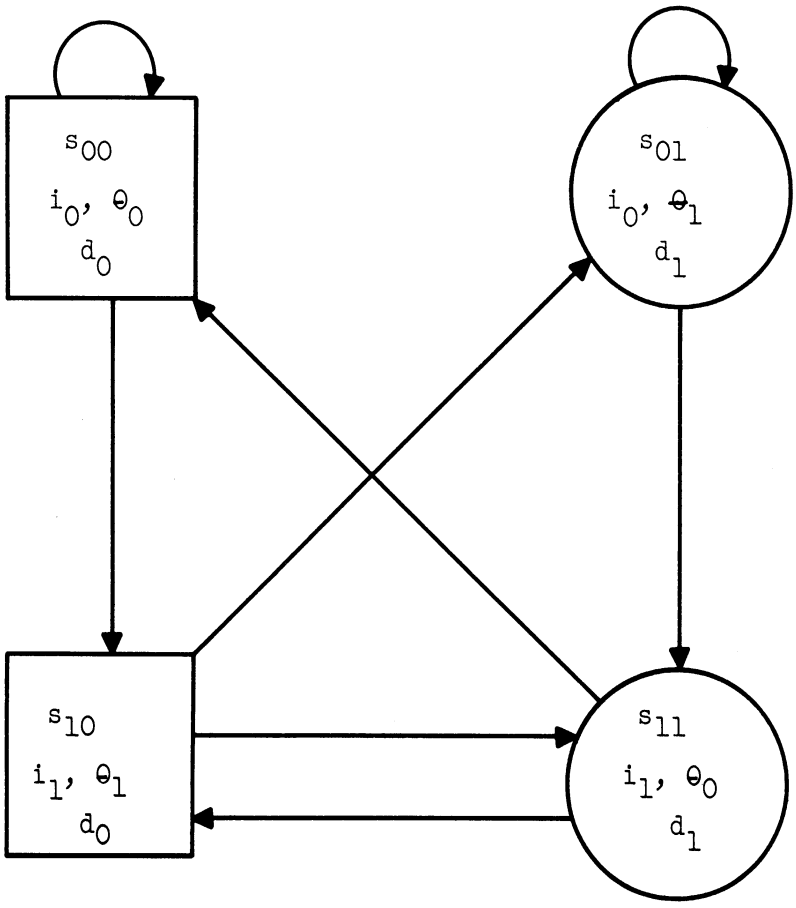
We mentioned in Section 1.2 that certain puzzles give rise to sequence generators. The so-called "15 puzzle" is a good example since it may be solved by means of our Reduced Form Algorithm. The puzzle consists of a 4 x 4 array of 15 movable blocks (number 1 through 15) and one empty position. A "move" consists in changing a pattern into any one of the (at most) four patterns obtained by shifting a block into the (neighboring) empty space. The problem is to achieve a stipulated pattern by a succession of moves starting from a given pattern. A sequence generator $\Gamma = (S,G,R,P)$ corresponding to the puzzle may be defined as follows. The 4 x 4 matrices whose entries are the numbers 0 through 15 are the complete states of Γ; there are 16! of these. The starting pattern is the sole generator of Γ. Two states s_1 and s_2 stand in the direct transition relation R if there is a move taking the pattern corresponding to s_1 into the pattern corresponding to s_2. The projection P has the value 1 on the single pattern stipulated

to be the goal and 0 otherwise. The problem is solved by constructing a finite Γ-sequence $\langle s(0), s(1), s(2),\ldots,s(t)\rangle$ such that $P[s(t)]= 1$ and $P[s(i)] = 0$ for $i<t$, if such a sequence exists. Clearly this sequence exists if and only if the complete state with a projection of 1 is Γ-accessible. Whether or not this is the case can be determined by applying part 2 of the Reduced Form Algorithm to Γ: if such a sequence exists, it will be found in the course of carrying out the algorithm. It turns out that exactly half of the complete states of Γ are Γ-accessible and that each of these Γ-accessible states is also Γ-admissible.

There is a much simpler algorithm for finding the Γ-accessible states of this particular sequence generator. See W. W. R. Ball, Mathematical Recreations and Essays, MacMillan, 1940, pp. 299-303.

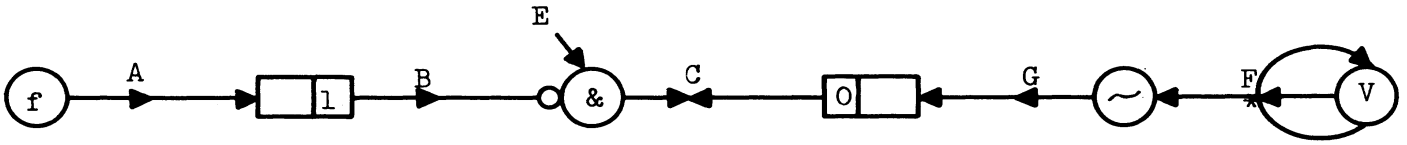


(a) Binary Counter

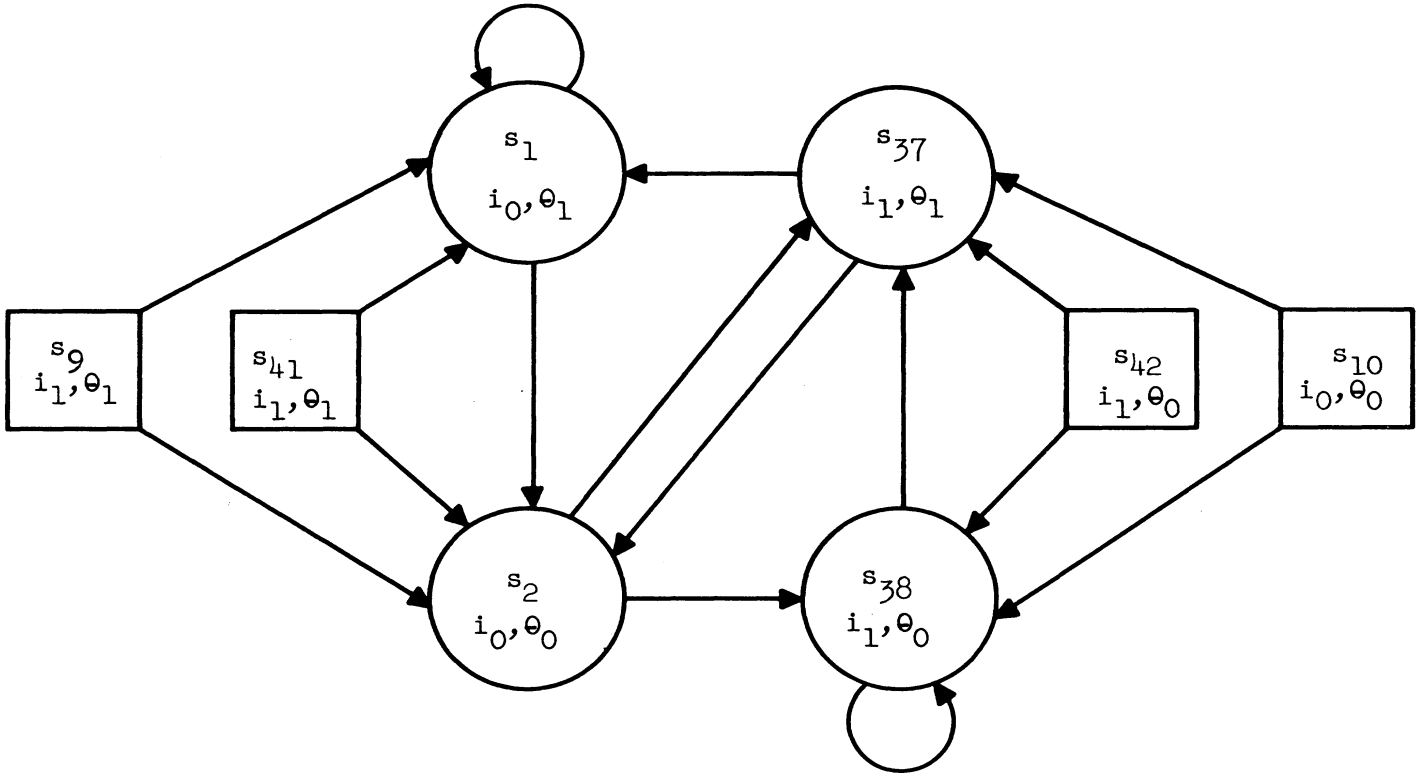


(b) Three-projection sequence generator $\Gamma = (S, G, R, I, \Theta, D)$ associated with the binary counter (a).

Figure 1.2-1

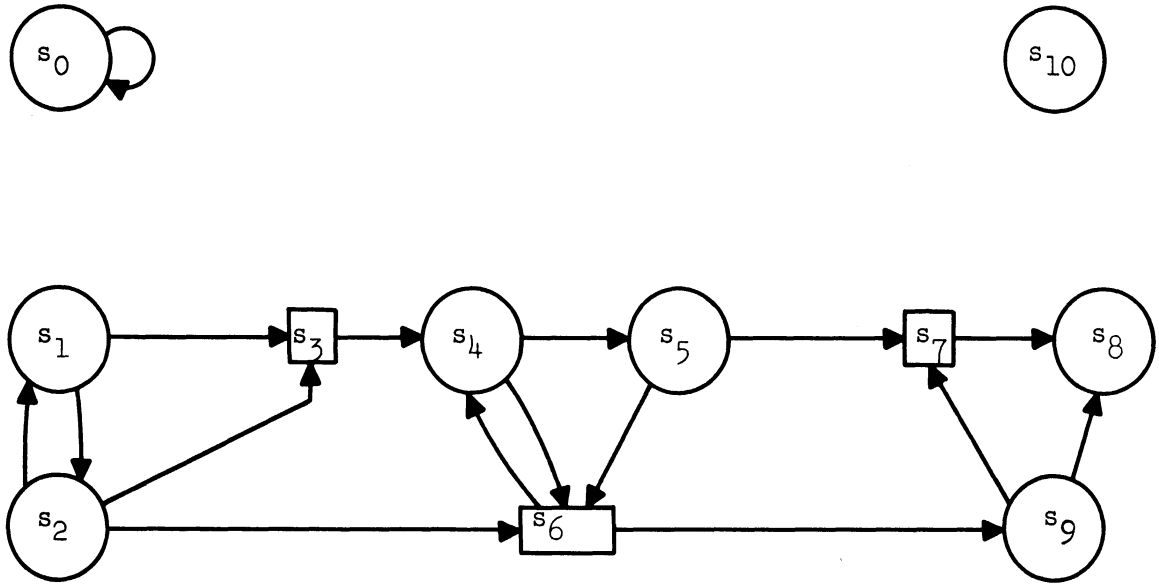


(a) Ill-formed net with behavior $F(t) \equiv \sim E(t+1)$

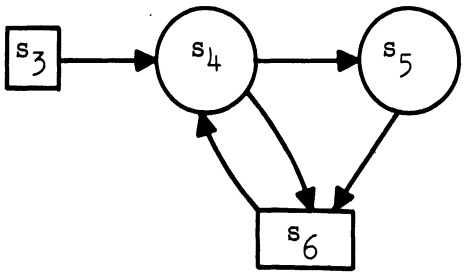


(b) Sequence generator $\Gamma = (S, G, R, I, \theta)$ associated with net (a)

Figure 1.2-2



(a) Sequence generator $\Gamma = (S, G, R)$



(b) Γ^+, the reduced form of Γ

Figure 1.3-1

2. Sequence Generators with One Projection

2.1 Definitions

In the last sub-section we made no particular use of the projections of a sequence generator. In this section we shall define some concepts which apply primarily to sequence generators with one projection and will prove some theorems about these concepts. In most applications this single projection is an input projection, an output projection, or a combined input-output projection. In the next section we will work with sequence generators having two projections. These two projections will usually be an input and an output projection.

Definition: The behavior of $\Gamma = (S, G, R, P^1, P^2, \ldots, P^n)$, where $n>0$, is the set $P([s](0,k))$, where $P = P^1 \times P^2 \times \ldots P^n$ and $[s](0,k)$ is a Γ-sequence (finite or infinite). "$B(\Gamma)$" denotes the behavior of Γ. The infinite behavior of Γ, denoted by "$B^\omega(\Gamma)$", is the set of infinite sequences in $B(\Gamma)$. Corollary 1.3-3b can now be reformulated as follows.

Corollary 2.1-1: $B^\omega(\Gamma) = B^\omega(\Gamma^+)$

It is worth noting that in general it is not true that for a sequence generator $\Gamma = (S,G,R,P)$ there exists a sequence generator $\dot{\Gamma}$ such that the set of $\dot{\Gamma}$-sequence equals the behavior of Γ. This may be shown by a simple example. Let $S = \{s_0, s_1, s_2, \}$, $G = \{s_0\}$, $R = \{\langle s_0, s_1\rangle, \langle s_1, s_2\rangle, \langle s_2, s_0\rangle\}$, and $P(s_0) = P(s_1) = p_0$, $P(s_2) = p_1$. There is one infinite Γ-sequence $\langle s_0, s_1, s_2, s_0, s_1, s_2, s_0, s_1, s_2 \ldots\rangle$ and the behavior of Γ consists of the sequence $\langle p_0, p_0, p_1, p_0, p_0, p_1, \ldots\rangle$ and all its initial segments, and does not include the infinite sequence $\langle p_0, p_0, p_0, \ldots\rangle$. Consider a sequence generator $\dot{\Gamma} = (\dot{S}, \dot{G}, \dot{R})$ such that $\{p_0, p_1\} \subset \dot{S}$ and such that $\langle p_0, p_0, p_1, p_0, p_0, p_1, \ldots\rangle$ is an infinite $\dot{\Gamma}$-sequence. It follows from the existence of this sequence that $\dot{R}(p_0, p_0)$ and $p_0 \in \dot{G}$, and hence that $\langle p_0, p_0, p_0, \ldots\rangle$ is an infinite $\dot{\Gamma}$-sequence.

We will now make some remarks about the application of the concept of behavior to nets. By the methods of Section 1.2 we can associate with every net (well-formed or not) a sequence generator $\Gamma = (S, G, R, I, \Theta)$, where I is the input projection and Θ is the output projection. The behavior of a digital computer (w.f.n.) consists of the relationship between its inputs and its outputs, and similarly for an arbitrary net. The behavior of a net may be regarded as the set of sequences (finite and infinite) of pairs $\langle i(0), \Theta(0)\rangle$, $\langle i(1), \Theta(1)\rangle$, $\langle i(2), \Theta(2)\rangle, \ldots$ for which there is a Γ-sequence $[s](0,k)$ such that

$i(t) = I\{s(t)\}$ and $\theta(t) = \theta\{s(t)\}$ for every t. This is clearly $B(\Gamma)$, the behavior of the sequence generator $\Gamma = (S, G, R, I, \theta)$. In Section 2.3 we present a Behavior Inclusion Procedure to be applied to a pair $\langle\Gamma, \dot{\Gamma}\rangle$ to decide whether the behavior of Γ is included in the behavior of $\dot{\Gamma}$; when applied to the pair $\langle\dot{\Gamma}, \Gamma\rangle$ as well as to the pair $\langle\Gamma, \dot{\Gamma}\rangle$ this tells us whether the behaviors of Γ and $\dot{\Gamma}$ are equal. By the remarks just made the Behavior Inclusion Procedure can be used to decide whether the behavior of an arbitrary net N is included in or equal to the behavior of a net $\dot{N}$. In the case of well-formed nets, however, a much more efficient algorithm for deciding equality of behaviors is known (Burks, Wang, 1956, Section 2.2); a basic part of this algorithm consists essentially of finding the reduced form (Section 1.3) of a sequence generator associated with the combined nets. Actually this algorithm applies to any deterministic sequence generator (this concept is defined below); moreover, if Γ and $\dot{\Gamma}$ are both deterministic and $B(\Gamma) \subset B(\dot{\Gamma})$, then $B(\dot{\Gamma}) \subset B(\Gamma)$, so this algorithm also answers the question as to whether $B(\Gamma) \subset B(\dot{\Gamma})$ for the case of deterministic sequence generators.

The following lemma will be needed in subsequent proofs. It is a classical interpretation of Brouwer's Fan theorem (Heyting, 1956, pp. 42-43) and is closely related to König's Infinity Lemma concerning infinite graphs (König, 1936, p. 81). Our lemma, however, is stronger than König's Infinity Lemma in that it does not require that the α's be pairwise disjoint; because of this difference we present here a proof of it.

Lemma 2.1-2: Let $\langle\alpha_0,\alpha_1,\alpha_2,\ldots\rangle$ be an ω-sequence of finite non-empty sets and let ρ be a binary relation. If for every $x \in \alpha_{i+1}$ there is a $y \in \alpha_i$ such that $\rho(y,x)$, then there is an infinite sequence $\langle z_0,z_1,z_2,\ldots\rangle$ such that for each i, $z_i \in \alpha_i$ and $\rho(z_i,z_{i+1})$.

Proof: Let β_i consist of all finite sequences $\langle x_i,x_{i+1},\ldots,x_{i+k}\rangle$ where $k = 0,1,2,\ldots, x_j \in \alpha_j$ for $i \leqq j \leqq i+k$ and $\rho(x_j, x_{j+1})$ for $i \leqq j < i+k$. It follows from the requirement on ρ in the hypothesis of the lemma that for each i,k, and element y_{i+k} of α_{i+k} there is an element of β_i $\langle y_i, y_{i+1},\ldots,y_{i+k}\rangle$. Since this is so for any k, each β_i is infinite. We will now define by induction the desired sequence $\langle z_0, z_1, z_2,\ldots\rangle$. Initial step: Since β_0 is infinite while α_0 is finite there will be some element z_0 of α_0 such that an infinite number of elements of β_0 begin with z_0. Let δ_0 be the subset of β_0 all of whose elements begin with z_0.

General step: Assume given a sequence $\langle z_0, z_1, \ldots, z_i \rangle$ (where $i = 0,1,2,\ldots$) which belongs to β_0 and satisfies the condition that the set δ_i of elements of β_i which begin with z_i is infinite. The result δ_{i+1} of deleting the first element of each member of β_i is an infinite subset of β_{i+1}. Since α_{i+1} is finite there will be some element z_{i+1} of α_{i+1} such that $\rho(z_i, z_{i+1})$ and an infinite number of elements of δ'_{i+1} begin with z_{i+1}. Let δ_{i+1} be the subset of δ'_{i+1} all of whose elements begin with z_{i+1}; δ'_{i+1} is a subset of β_{i+1} and hence δ_{i+1} is also. Hence $\langle z_0, z_1, \ldots, z_i, z_{i+1} \rangle$ belongs to β_0 and satisfies the condition that the set δ_{i+1} of elements of β_{i+1} which begin with z_{i+1} is infinite. Thus the inductive hypothesis has been established for the sequence $\langle z_0, z_1, \ldots, z_i, z_{i+1} \rangle$. This completes the proof of Lemma 2.1-2.

Theorem 2.1-3 (Infinity Theorem): Let $\Gamma = (S, G, R, P)$ be a sequence generator with behavior $B(\Gamma)$ and let $[p](0,\omega)$ be an infinite sequence of P-states. $[p](0,\omega) \in B(\Gamma)$ if and only if for every finite k, $[p](0,k) \in B(\Gamma)$.

Proof: (I) It is obvious that $[p](0,\omega) \in B(\Gamma)$ implies that for every finite k, $[p](0,k) \in B(\Gamma)$. (II) Suppose that for every k, $[p](0,k) \in B(\Gamma)$. We will prove that $[p](0,\omega) \in B(\Gamma)$ by means of Lemma 2.1.2. We define α_i by $s_1 \in \alpha_i$ if there exists a Γ-sequence $[s](0,i)$ such that $[p](0,i) = P\{[s](0,i)\}$ and $s_i = s(i)$. It is clear that each α_i is finite and non-empty. We let $\rho = R$ and show that the hypothesis of Lemma 2.1.2 is satisfied. Suppose $s_2 \in \alpha_{i+1}$. By definition of α_{i+1} there exists a Γ-sequence $[s_3](0,i+1)$ such that $[p](0,i+1) = P\{[s_3](0, i+1)\}$ and $s_2 = [s_3](i+1)$. Let $s_4 = s_3(i)$. By the definition of α_i, $s_4 \in \alpha_i$ and by the definition of Γ-sequence, $R(s_1, s_2)$. By Lemma 2.1.2 there is an infinite Γ-sequence $[s_5](0,\omega)$ and by the definition of α_i we have $P\{[s_5](0,\omega)\} = [p](0,\omega)$. Hence $[p](0,\omega) \in B(\Gamma)$.

The following is a corollary of the Infinity Theorem. Let $\Gamma = (S, G, R, P)$ and $\dot{\Gamma} = (\dot{S}, \dot{G}, \dot{R}, \dot{P})$ be two sequence generators and suppose that for every finite k, if $[p](0,k) \in B(\Gamma)$ then $[p](0,k) \in B(\dot{\Gamma})$; then $B(\Gamma) \subset B(\dot{\Gamma})$. For consider any infinite sequence $[p](0,\omega) \in B(\Gamma)$. By the Infinity Theorem, for each k $[p](0,k) \in B(\Gamma)$. Then by hypothesis, for each k $[p](0,k) \in B(\dot{\Gamma})$. Finally, by the Infinity Theorem $[p](0,\omega) \in B(\dot{\Gamma})$. This result holds for Γ and $\dot{\Gamma}$ interchanged, of course, so we have: if for every finite k $[p](0,k) \in B(\Gamma) \equiv [p](0,k) \in B(\dot{\Gamma})$ then $\beta(\Gamma) = \beta(\dot{\Gamma})$. Thus the Infinity Theorem shows that the "finite" behavior of a sequence generator determines its (complete) behavior.

Definitions: Let $\Gamma = (S, G, R, P)$ be a sequence generator. Γ is solvable if every infinite sequence of P-states belongs to its behavior. Γ is {semi-deterministic} [deterministic] if it satisfies the conditions: (1) For any P-state p, there is {at most one} [exactly one] complete state s of Γ such that $s \in G$ and $P(s) = p$. (2) For any complete state s_1 and any P-state p of Γ, there is {at most one} [exactly one] complete state s_2 such that $R(s_1, s_2)$ and $P(s_2) = p$.

It is obvious from the definition of {semi-determinism} [determinism] that there is a decision procedure for the class of {semi-deterministic} [deterministic] sequence generators. The problem of solvability is not so simple, but we will later develop a decision procedure for solvability (see Theorem 2.3-2).

Let us illustrate these concepts. The sequence generator of Figure 1.2-1b, less its last two projections, is clearly deterministic. The sequence generator of Figure 2.1-1a is semi-deterministic but not solvable, while the sequence generator of Figure 2.1-1b is neither semi-deterministic nor solvable. By simple inspection it can be ascertained that the sequence generator (S,G,R,P) of Figure 3.2-2a is neither semi-deterministic nor deterministic. It is, however, solvable, as the following considerations show. Given any sequence of P-states divide it into a sequence (finite or infinite) of subsequences (finite or infinite), where each subsequence is either an iteration of p_0 or an iteration of p_1 and the two types of subsequences alternate. Now a Γ-sequence $s_0,s_0,\ldots,s_0,s_1$ produces a P-state sequence $p_0, p_0,\ldots,p_0,p_0$ followed by at least one occurrence of p_1, while a Γ-sequence $s_3,s_3,\ldots,s_3,s_2$ produces a P-state sequence $p_1,p_1,\ldots,p_1,p_1$ followed by at least one occurrence of p_0. Hence for any sequence of P-states [p](0,k) one can construct a Γ-sequence [s](0,k) such that [p](0,k) = p([s](0,k)), and so (S,G,R,P) is solvable. Consider next (S,G,R,I) of Figure 1.2-2b. (S,G,R,I) is not semi-deterministic, since the input sequence $\langle i_0,i_0,i_0\rangle$ is the projection of both $\langle s_9,s_1,s_1\rangle$ and $\langle s_9,s_1,s_2\rangle$. But (S,G,R,I) is solvable, as may be shown by an analysis like that just given for Figure 3.2-2a; indeed, except for labeling, the behavior of Figure 1.2-2b is the same as the behavior of Figure 3.2-2a.

The following lemma may be established by simple mathematical inductions with reference to the appropriate definitions.

Lemma 2.1-4: Let Γ = (S, G, R, P). (a) If Γ is {semi-deterministic} [deterministic] (solvable) then Γ^+ is {semi-deterministic} [deterministic] (solvable). (b) If every complete state of Γ is Γ-accessible, then Γ is {semi-deterministic} [deterministic] if and only if for every finite sequence of P-states [p](0,t) there exists {at most one} [exactly one] Γ-sequence [s](0,t) such that P{[s](0,t)} = [p](0,t). (c) If Γ is deterministic, then Γ is solvable.

Other senses of semi-determinism and of determinism may be obtained by replacing every occurrence of "complete state" in the above defintion of semi-determinism and determinism either by "admissible

complete state" or by "accessible complete state". We will call the concepts obtained by making the latter substitution "semi-determinism$_1$" and "determinism$_1$". It may be shown that these two concepts are equivalent to the conditions stated in the consequent of part (b) of Lemma 2.1-4. In the case of arbitrary nets determinism$_1$ becomes the determinism of Burks and Wright, 1953, p. 1359.

The process described in Section 1.2 associates with a finite automaton a sequence generator $\Gamma = (S, G, R, I, \Theta, D)$ such that (S, G, R, I) is deterministic. Conversely, given a sequence generator $\Gamma = (S, G, R, I, \Theta)$, where (S, G, R, I) is deterministic, we can define a corresponding finite automaton. Let the set of input states $\{i\}$ and the set of output states $\{\theta\}$ be the ranges of the projections I and Θ respectively. The set of internal states $\{d\}$ of the sutomaton is a set of sets of complete states of Γ defined as follows: $\alpha \in \{d\} .\equiv. \alpha = G .v. (\exists s)\ (\alpha = R(s))$, where α ranges over non-null subsets of S. Let the initial internal state $d_0 = G$. The direct transition function τ is given by $\tau(i,d) = R(\imath s \mid s \in d \ \& \ I(s) = i)$, where "$\imath s \mid$ ..." means "the complete state s satisfying the condition"...". Finally, the output function λ of the net is defined by $\lambda(i,d) = \Theta(\imath s \mid s \in d \ \& \ I(s) = i)$. We will give an example. Figure 2.1-2a is a deterministic sequence generator. The set of input states for the associated automaton is $\{i_0, i_1\}$ and the set of output states is $\{\Theta_0, \Theta_1, \Theta_2, \Theta_3, \Theta_4\}$. The set of internal states consists of the sets $\{s_0, s_1\}$, $\{s_2, s_3\}$, and $\{s_3, s_4\}$, which we will call d_0, d_1, and d_2 respectively. d_0 is the initial internal state since $\{s_0, s_1\} = G$. The direct transition and output functions are given by the table

$i(t)$	$d(t)$	$d(t+1) = \tau(i,d)$	$\theta(t) = \lambda(i,d)$
i_0	d_0	d_1	θ_0
i_1	d_0	d_2	θ_1
i_0	d_1	d_0	θ_2
i_1	d_1	d_0	θ_3
i_0	d_2	d_0	θ_4
i_1	d_2	d_0	θ_3

In Section 1.2 we gave a process for converting a finite automaton into a three projection sequence generator. When this process is applied to the finite automaton just described the result is the sequence generator $\dot{\Gamma} = (\dot{S}, \dot{G}, \dot{R}, \dot{I}, \dot{\Theta}, \dot{D})$ of Figure 2.1-2b. It should be noted that $B(\Gamma) = B(\dot{S}, \dot{G}, \dot{R}, \dot{I}, \dot{\Theta})$. Hence when the two procedures just described are applied successively to a sequence generator $\Gamma = (S, G, R, I, \Theta)$,

where (S, G, R, I) is deterministic, the result is a sequence generator $\dot{\Gamma} = (\dot{S}, \dot{G}, \dot{R}, \dot{I}, \dot{\Theta}, \dot{D})$ with an internal state projection $\dot{D}$ and such that the behavior of $(\dot{S}, \dot{G}, \dot{R}, \dot{I}, \dot{\Theta})$ is the same as the behavior of Γ.

Any property of a one-projection sequence generator and any operation applicable to a one-projection sequence generator can be extended to a sequence generator $\Gamma = (S, G, R)$ without projections by adjoining to it a constant projection P (i.e., a projection with only one P-state); Γ is solvable, deterministic, etc. if (S, G, R, P) is solvable, deterministic, etc. For example, a well-formed net without input nodes has associated with it (by either of the techniques of Section 1.2) a sequence generator (S, G, R) with one infinite (periodic) Γ-sequence. For constant P, (S, G, R, P) is solvable and semi-deterministic, and hence deterministic, and so is (S, G, R); see, for example, Figure 2.2-1a.

2.2 Subset sequence generator operation

We will next define an operation, denoted by "*", called "the subset sequence generator operation". This operation may be applied to any sequence generator Γ to obtain its subset sequence generator Γ^*. The complete states of Γ^* are sets of complete states of Γ. The generators, the direct transition relation, and the projections of Γ^* are defined in terms of Γ in such a way that Γ^* has the same behavior as Γ (Theorem 2.2-3 below) and Γ^* is always semi-deterministic, even though Γ may not be (Lemma 2.2-1 below).

Definition: The subset sequence generator operation, denoted by "*", applies to any sequence generator $\Gamma = (S, G, R, P^1, P^2, \ldots, P^n)$, where $n>0$, and produces a sequence generator $\Gamma^* = \dot{\Gamma} = (\dot{S}, \dot{G}, \dot{R}, \dot{P}^1, \dot{P}^2, \ldots, \dot{P}^n)$. Let $P = P^1 \times P^2 \times \ldots \times P^n$. (1) A subset x of S is an element of $\dot{S}$ if and only if x is non-null and P has the same value for all elements of x. This definition can be expressed symbolically as follows, where Λ is the null set, and the variable x ranges over subsets of S:

$$x \in \dot{S} :\equiv: x \neq \Lambda \;\&\; (s_1, s_2)\{[(s_1 \in S) \;\&\; (s_2 \in S) \;\&\; (s_1 \in x) \;\&\; (s_2 \in x) \supset [P(s_1) = P(s_2)]]\}$$

(2) The elements of $\dot{G}$ are maximal subsets of G which are elements of $\dot{S}$. Formally, $\dot{s} \in \dot{G} :\equiv: \dot{s} \in \dot{S} \;\&\; \dot{s} \subset G \;\&\; (\dot{s}_1)\{[\dot{s}_1 \in \dot{S} \;\&\; (\dot{s} \subset \dot{s}_1 \subset G)] \supset (\dot{s} = \dot{s}_1)\}$

(3) Two complete states $\dot{s}_1$ and $\dot{s}_2$ of $\dot{S}$ stand in the direct transition relation $\dot{R}$ if and only if $\dot{s}_2$ is a maximal set of direct successors (by R) of elements of $\dot{s}_1$. Formally,

$$\dot{R}(\dot{s}_1, \dot{s}_2) :\equiv: \dot{s}_1 \in \dot{S} \;\&\; s_2 \subset R(\dot{s}_1) \;\&\; (\dot{s}_3)\{[\dot{s}_3 \in \dot{S} \;\&\; (\dot{s}_2 \subset \dot{s}_3 \subset R(\dot{s}_1))] \supset (\dot{s}_2 = \dot{s}_3)\}$$

(4) All the elements of a state $\dot{s}$ of $\dot{S}$ have the same P^i state (for $i = 1,2,\ldots,n$), and we take this common-value to be the $\dot{P}^i$-state of $\dot{s}$. Formally, $\dot{P}^i(\dot{s}) = P^i(s)$ where $s \in \dot{s}$, $\dot{s} \in \dot{S}$.
(Γ^* is called "the subset sequence generator" of Γ. Our concept of a subset sequence generator is similar to concepts used by Myhill, 1957, p. 122, Medvedev, 1958, p. 13, and Rabin and Scott, 1959, Definition 11.)

It should be noted that the concepts of behavior (Section 2.1) and subset sequence generator are essentially one-projection concepts in the sense that when many projections $P^1, P^2, \ldots P^n$ are given the composite projection $P^1 x P^2 x \ldots P^n$ is used in the definitions of the concepts. In subsequent theorems and algorithms we will, for the sake of simplicity, usually state our results for sequence generators with one projection, since it is obvious how to extend them to the many-projection case.

The construction of subset sequence generators is illustrated in Figures 2.2-1, 2.2-2. Note that the generators and complete states of the subset sequence generator Γ^* are determined without reference to the direct transition relation of Γ. Figure 2.2-1 shows that even though Γ is in reduced form, Γ^* may not be. Though in fact, if Γ is in reduced form then Γ^* has no terminal states and so $B(\Gamma^*) = B(\Gamma^{*+})$. In Figure 2.2-2 we begin with a semi-deterministic sequence generator, add to it in various ways to obtain three sequence generators, $\dot{\Gamma}$, $\ddot{\Gamma}$, $\dddot{\Gamma}$ which are not semi-deterministic, and then derive the subset sequence generator of each of these. All the subset sequence generator Γ^*, $\dot{\Gamma}^*$, $\ddot{\Gamma}^*$, $\dddot{\Gamma}^*$ are semi-deterministic, as they must be by the next lemma. None of the sequence generators Γ, $\dot{\Gamma}$, $\ddot{\Gamma}$, $\dddot{\Gamma}$ is solvable; Γ^*, $\dot{\Gamma}^*$, $\ddot{\Gamma}^*$, and $\dddot{\Gamma}^*$ are not solvable either (cf. corollary 2.2-4).

Lemma 2.2-1: For any sequence generator $\Gamma = (S, G, R, P)$, Γ^* is semi-deterministic.

Proof: Let $\Gamma^* = \dot{\Gamma} = (\dot{S}, \dot{G}, \dot{R}, \dot{P})$. It follows from the construction of $\dot{G}$ that for any $\dot{s}_1$, $\dot{s}_2$, if $\dot{s}_1 \in \dot{G}$, $\dot{s}_2 \in \dot{G}$, and $\dot{P}(\dot{s}_1) = \dot{P}(\dot{s}_2)$, then $\dot{s}_1 = \dot{s}_2$, and it follows from the definition of R that for any $\dot{s}$, $\dot{s}_1$, $\dot{s}_2$, if $\dot{R}(\dot{s},\dot{s}_1)$, $\dot{R}(\dot{s},\dot{s}_2)$, and $\dot{P}(\dot{s}_1) = \dot{P}(\dot{s}_2)$ then $\dot{s}_1 = \dot{s}_2$.

Given a sequence generator $\Gamma = (S, G, R, P)$, by the above lemma its subset sequence generator $\Gamma^* = \dot{\Gamma} = (\dot{S}, \dot{G}, \dot{R}, \dot{P})$ is semi-deterministic. Hence for any given finite sequence of P-states $[p](0,t)$ there is at most one complete state $\dot{s}_1$ satisfying the condition that there exists a $\dot{\Gamma}$-sequence $[\dot{s}](0,t-1)$, $\dot{s}_1$ such that $P\{[\dot{s}](0,t-1), \dot{s}_1\} = [p](0,t)$. Moreover, this state

$\mathring{s}_1$ is a set of states of Γ. We will use the location "the state $\mathring{s}_1 \in \mathring{S}$ corresponding to $[p](0,t)$" to refer to this set of states $\mathring{s}_1$ if it exists, otherwise to the null set.

Lemma 2.2-2: Let $\Gamma^* = \mathring{\Gamma} = (\mathring{S}, \mathring{G}, \mathring{R}, \mathring{P})$ be the subset sequence generator of $\Gamma = (S, G, R, P)$, let $[p](0,t)$ be any finite sequence of P-states, and let α be the set of states s_1 for which there exists a Γ-sequence $[s](0,t-1), s_1$ such that $P\{[s](0,t-1), s_1\} = [p](0,t)$. Then α is the state $\mathring{s}_1 \in \mathring{S}$ corresponding to $[p](0,t)$.

Proof: (I) We first prove by an induction on t that for any finite Γ-sequence $[s](0,t)$ there is a $\mathring{\Gamma}$-sequence $[\mathring{s}](0,t)$ satisfying the conditions

$$(a) \quad P\{[s](0,t)\} = \mathring{P}\{[\mathring{s}](0,t)\}$$

$$(b) \quad s(t) \in \mathring{s}(t).$$

Initial step: given the Γ-sequence $s(0)$, it follows by the definition of $\mathring{G}$ that there exists a complete state $\mathring{s}(0)$ such that $s(0) \in \mathring{s}(0)$ and $\mathring{s}(0) \in \mathring{G}$. General step: The inductive hypothesis is that for every Γ-sequence $[\mathring{s}](0,k)$ there is a $\mathring{\Gamma}$-sequence $[s](0,k)$ satisfying the conditions (a) and (b). Consider any Γ-sequence $[s](0,k+1)$. By the inductive hypothesis there exists a $\mathring{\Gamma}$-sequence $[\mathring{s}](0,k)$ such that $[s](0,k)$ and $[\mathring{s}](0,k)$ satisfy (a) and (b). Since $R(s(k), s(k+1))$ it follows by the definition of $\mathring{R}$ that there exists a complete state $\mathring{s}_1$ such that $s(k+1) \in \mathring{s}_1$ and $\mathring{R}(\mathring{s}(k), \mathring{s}_1)$. Hence $P\{s(k+1)\} = \mathring{P}(\mathring{s}_1)$ and $[\mathring{s}](0,k), \mathring{s}_1$ is a Γ-sequence, and so $[s](0,k+1)$ and $[\mathring{s}](0,k), \mathring{s}_1$ satisfy conditions (a) and (b). (II) We next prove by an induction on t that for any finite $\mathring{\Gamma}$-sequence $[\mathring{s}](0,t)$ and complete state $s_1 \in \mathring{s}(t)$ there is a Γ-sequence $[s](0,t)$ satisfying the conditions

$$(c) \quad P\{[s](0,t)\} = \mathring{P}\{[\mathring{s}](0,t)\}$$

$$(d) \quad s(t) = s_1$$

Initial step: given a $\mathring{\Gamma}$-sequence $\mathring{s}(0)$ and a state $s_1 \in \mathring{s}(0)$, it follows by the definition of $\mathring{G}$ that s_1 is the desired Γ-sequence. General step: the inductive hypothesis is that for every $\mathring{\Gamma}$-sequence $[\mathring{s}](0,k)$ and state $s_1 \in \mathring{s}(k)$ there is a Γ-sequence $[s](0,k)$ satisfying conditions (c) and (d). Consider any $\mathring{\Gamma}$-sequence $[\mathring{s}](0,k+1)$ and a state $s_2 \in \mathring{s}(k+1)$. By the definition of $\mathring{R}$ there exists a state s_1 satisfying the conditions $s_1 \in \mathring{s}(k)$ and $R(s_1,s_2)$. By the inductive hypothesis there is a Γ-sequence $[s](0,k)$ such that $P\{[s](0,k)\} = \mathring{P}\{[\mathring{s}](0,k)\}$ and $s(k) = s_1$. $[s](0,k), s_2$ is the desired Γ-sequence. This completes the proof of Lemma 2.2-2.

Theorem 2.2-3: For any sequence generator $\Gamma = (S, G, R, P)$ with behavior $B(\Gamma)$, $B(\Gamma) = B(\Gamma)^*$.

Proof: By the preceding Lemma 2.2-2, for every finite t $[p](0,t) \in B(\Gamma)$ if and only if $[p](0,t) \in B(\Gamma^*)$. The theorem to be proved now follows by the Infinity Theorem (2.1-3). It should be noted in this connection that the proof of the Infinity Theorem makes implicit use of Γ^*, the subset sequence generator of Γ. In fact, the sequence $\langle\alpha_0,\alpha_1,\alpha_2,\ldots\rangle$ employed in the proof is an infinite Γ^*-sequence.

Lemma 2.2-4: For any sequence generator $\Gamma = (S, G, R, P)$, Γ is solvable if and only if Γ^* is solvable. (This follows easily from part 2 of Lemma 2.1-4, Lemma 2.2-1, and Theorem 2.2-3.)

2.3 Decision Procedures

Behavior Inclusion Procedure: Consider two sequence generators $\Gamma = (S, G, R, P)$ and $\dot{\Gamma} = (\dot{S}, \dot{G}, \dot{R}, \dot{P})$, and let $\{a\}$ $[\dot{a}]$ be the number of states in $\{S\}$ $[\dot{S}]$. Form all $\{\Gamma\text{-sequences}\}$ $[\dot{\Gamma}\text{-sequences}]$ of length $1 + a2^{\dot{a}}$ or less and form the set $\{\alpha\}$ $[\dot{\alpha}]$ of their $\{P\text{-projections}\}$ $[\dot{P}\text{-projections}]$. Write "yes" or "no" as $\alpha \subset \dot{\alpha}$ or not.

Theorem 2.3-1: Let A be the class of pairs of one-projection sequence generators $\langle\Gamma,\dot{\Gamma}\rangle$ such that $B(\Gamma) \subset B(\dot{\Gamma})$, i.e., such that the behavior of Γ is included in that of $\dot{\Gamma}$. The Behavior Inclusion Procedure is a decision procedure for A.

Proof: (I) It is obvious that if $B(\Gamma) \subset B(\dot{\Gamma})$ then $\alpha \subset \dot{\alpha}$, i.e., that the algorithm yields "yes". (II) We assume that $\alpha \subset \dot{\alpha}$, i.e., that the algorithm yields "yes", and prove that $B(\Gamma) \subset B(\dot{\Gamma})$. Let $\ddot{\Gamma} = (\ddot{S}, \ddot{G}, \ddot{R}, \ddot{P}) = \dot{\Gamma}^*$ and let $\ddot{\alpha}$ be the set of $\ddot{P}$-projections of $\ddot{\Gamma}$-sequences of length $1 + a2^{\dot{a}}$ or less. By Theorem 2.2-3 $\dot{\alpha} = \ddot{\alpha}$ and $B(\dot{\Gamma}) = B(\ddot{\Gamma})$, so we will assume that $\alpha \subset \ddot{\alpha}$ and prove that $B(\Gamma) \subset B(\ddot{\Gamma})$. (IIA) Let $\{\alpha_t\}[\ddot{\alpha}_t]$ be the subset of sequences of $\{B(\Gamma)\}$ $[B(\ddot{\Gamma})]$ of length t+1 or less. We will now establish by induction that for every t, $\alpha_t \subset \ddot{\alpha}_t$.

Initial Step: Since by assumption $\alpha \subset \ddot{\alpha}$ and by definition $\alpha = \alpha_{a.2^{\dot{a}}}$ and $\ddot{\alpha} = \ddot{\alpha}_{a.2^{\dot{a}}}$ so $\alpha_k \subset \ddot{\alpha}_k$ for $k = a.2^{\dot{a}}$. General step: we assume that $\alpha_k \subset \ddot{\alpha}_k$ for $k \geq a.2^{\dot{a}}$ and prove that $\alpha_{k+1} \subset \ddot{\alpha}_{k+1}$. Consider an arbitrary Γ-sequence $[s](0,k+1)$ for $k \geq a.2^{\dot{a}}$. Let $[p](0,k+1) = P\{[s]0, k+1)]\}$. By the inductive hypothesis there exists a $\ddot{\Gamma}$-sequence $[s](0,k)$ such that $\ddot{P}\{[\ddot{s}](0,k)\} = [p](0,k)$.

We will show that there exists a complete state s_0 satisfying the conditions

(1) $R\{s(k), s_0\}$

(2) $P(s_0) = p(k+1)$,

i.e., that there exists a Γ-sequence $<[s](0,k), s_0>$ such that $P\{[s](0,k), s_0\} = [p](0,k+1)$. It follows from the nature of the subset sequence generator construction that Γ has no more than 2^a complete states, and hence the number of pairs of complete states $<s,s>$ is no more than $a.2^a$. Since $k \geq a.2^a$ there exist t_1, t_2 such that $0 \leq t_1 < t_2 \leq a2^a \leq k$, $s(t_1) = s(t_2)$ and $s(t_1) = s(t_2)$. Let

(3) $[s_1](0,\ell+1) = \{[s](0,t_1), [s](t_2+1,k+1)\}$

(4) $[s_1](0,\ell) = \{[s](0,t_1), [s](t_2+1,k)\}$

(5) $[p_1](0,\ell+1) = \{[p](0,t_1), [p](t_2+1, k+1)\}$

where $\ell = k - (t_2 - t_1)$. Note that

(6) $P\{[s_1](0,\ell+1)\} = [p_1](0,\ell+1)$

(7) $P\{[s_1](0,\ell)\} = [p_1](0,\ell)$

Because $\{s(t_1) = s(t_2)\}$ $[s(t_1) = s(t_2)]$ we have that $\{[s_1](0,\ell+1)$ is a Γ-sequence$\}$ $[[s_1](0,\ell)$ is a Γ-sequence$]$. And since $\ell + 1 \leq k$ there exists by the inductive hypothesis a Γ-sequence $[s_2](0,\ell+1)$ such that

(8) $P\{[s_2](0,\ell+1)\} = [p_1](0,\ell+1)$.

It follows from (7) and (8) that

(9) $P\{[s_2](0,\ell)\} = P\{[s_1](0,\ell)\} = [p_1](0,\ell)$

and since Γ is semi-deterministic (Lemmas 2.1-4 and 2.2-1) we have that

(10) $[s_2](0,\ell) = [s_1](0,\ell)$.

It follows from (4) that

(11) $s_1(\ell) = s(k)$

and hence by (10)

$$(12)\quad \ddot{s}_2(\ell) = \ddot{s}(k).$$

Since $[\ddot{s}_2](0, \ell+1)$ is a $\ddot{\Gamma}$-sequence we have that $\ddot{R}\{\ddot{s}_2(\ell), \ddot{s}_2(\ell+1)\}$ and hence by (12) that

$$(13)\quad \ddot{R}\{\ddot{s}(k), \ddot{s}_2(\ell+1)\}.$$

Now by (8) and (5)

$$(14)\quad \ddot{P}\{\ddot{s}_2(\ell+1)\} = p(k+1).$$

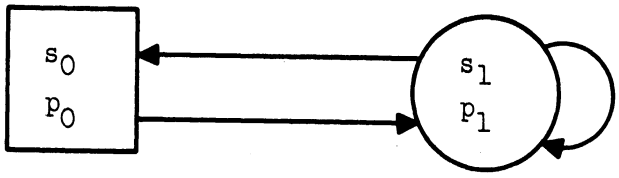
Conditions 13 and 14 show that $\ddot{s}_2(\ell+1)$ satisfies conditions (1) and (2) and hence that $\ddot{s}_2(\ell+1)$ is the desired state $\ddot{s}_0$. (IIB) We have shown that if $\alpha \dot{c} \dot{\alpha}$ then for every t, $\alpha_t \ddot{c} \ddot{\alpha}_t$. It follows by the Infinity Theorem (Theorem 2.1-3) that if $\alpha \dot{c} \dot{\alpha}$ then $B(\Gamma) \subset B(\ddot{\Gamma})$. As remarked earlier this is equivalent to: if $\alpha \dot{c} \dot{\alpha}$ then $B(\Gamma) \subset B(\dot{\Gamma})$. This completes the proof of Theorem 2.3-1.

In formulating the Behavior Inclusion Procedure we have not attempted to minimize the computation required. Many simplifications will occur to anyone who uses this algorithm. For example, since any two elements of a complete state $\ddot{s} \in \ddot{S}$ must have the same projection the bound $1 + a \cdot 2^{\dot{a}}$ may be greatly reduced. Note also that if $\dot{\Gamma}$ is already semi-deterministic, it is not necessary to make use of Γ^* in the proof, and the bound $1 + a \cdot 2^{\dot{a}}$ may be replaced by $1 + a\dot{a}$.

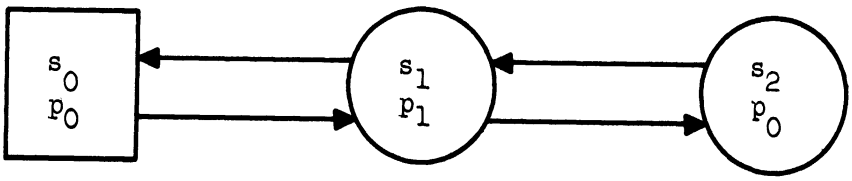
The Behavior Inclusion Procedure may be used as the basis of a decision procedure for solvability. Let $\Gamma = (S, G, R, P)$ be given. By definition Γ is solvable if every infinite sequence of P-states belongs to its behavior (Section 2.1). $\dot{\Gamma} = (S, S, \dot{R}, P)$, where $\dot{R}(s_1, s_2)$ for all $s_1, s_2 \in S$, has as its behavior the set of all sequences of P-states. Hence the behavior of $\dot{\Gamma}$ includes all infinite sequences of P-states, so Γ is solvable if and only if $B(\dot{\Gamma}) \subseteq B(\Gamma)$. By Theorem 2.3-1 the Behavior Inclusion Procedure is a decision procedure for behavior inclusion, so we have proved the following theorem.

Theorem 2.3-2: Let $\Gamma = (S, G, R, P)$ be a sequence generator and let $\dot{R}(s_1, s_2)$ for all $s_1, s_2 \in S$. The Behavior Inclusion Procedure applied to the pair $\langle (S, S, \dot{R}, P), \Gamma \rangle$ is a decision procedure for the solvability of Γ.

When the Behavior Inclusion Procedure is applied first to the pair $\langle \Gamma, \dot{\Gamma} \rangle$ and then to the pair $\langle \dot{\Gamma}, \Gamma \rangle$ the result is "yes" in both cases if and only if $B(\Gamma) = B(\dot{\Gamma})$. This "behavior equivalence procedure" may be used to reduce the number of complete states of a sequence generator so as to obtain a behaviorally equivalent sequence generator with fewer states. Consider $\Gamma = (S, G, R, P)$. We will say that two complete states s_1 and s_2 are "behaviorally equivalent" if $B[(S,\{s_1\}, R, P)] = B[(S,\{s_2\}, R, P)]$. Let $\dot{\Gamma}$ be the result of identifying all behaviorally equivalent states of Γ. $\dot{\Gamma}$ will in general have fewer states than Γ and yet $B(\dot{\Gamma}) = B(\Gamma)$. Moore's concept of two sequential machines being indistinguishable by any experiment is a special case of our concept of behavioral equivalence, and the above process of identifying behaviorally equivalent states is analogous to the "reduction procedure" of Moore, 1956 and Mealy, 1955. We have examples to show that the procedure we described does not always lead to a behaviorally equivalent sequence generator with a minimal number of complete states and that the procedure of Moore and Mealy does not always lead to a behaviorally equivalent sequential machine with a minimum number of internal states.

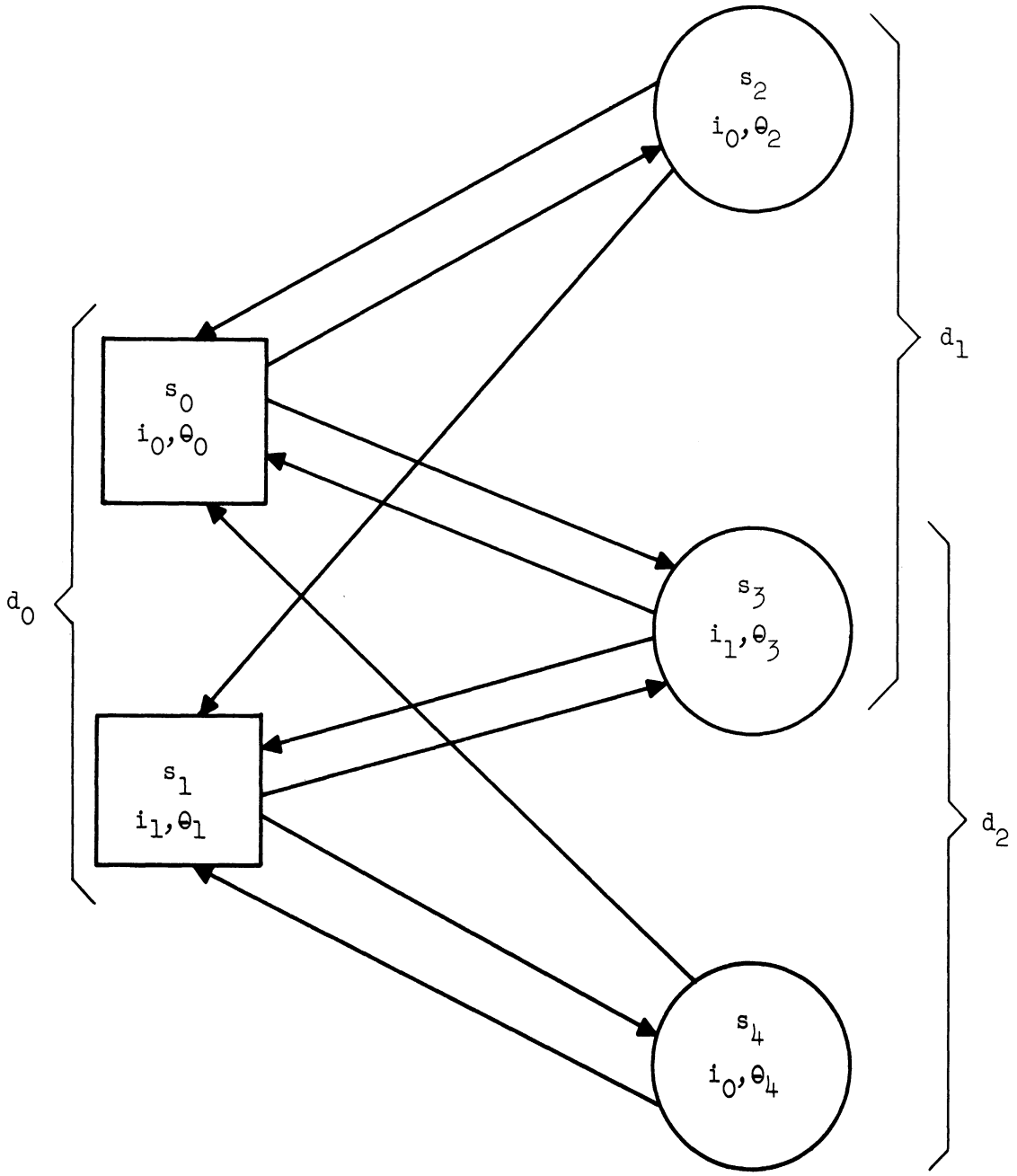


(a) Semi-deterministic non-solvable sequence generator



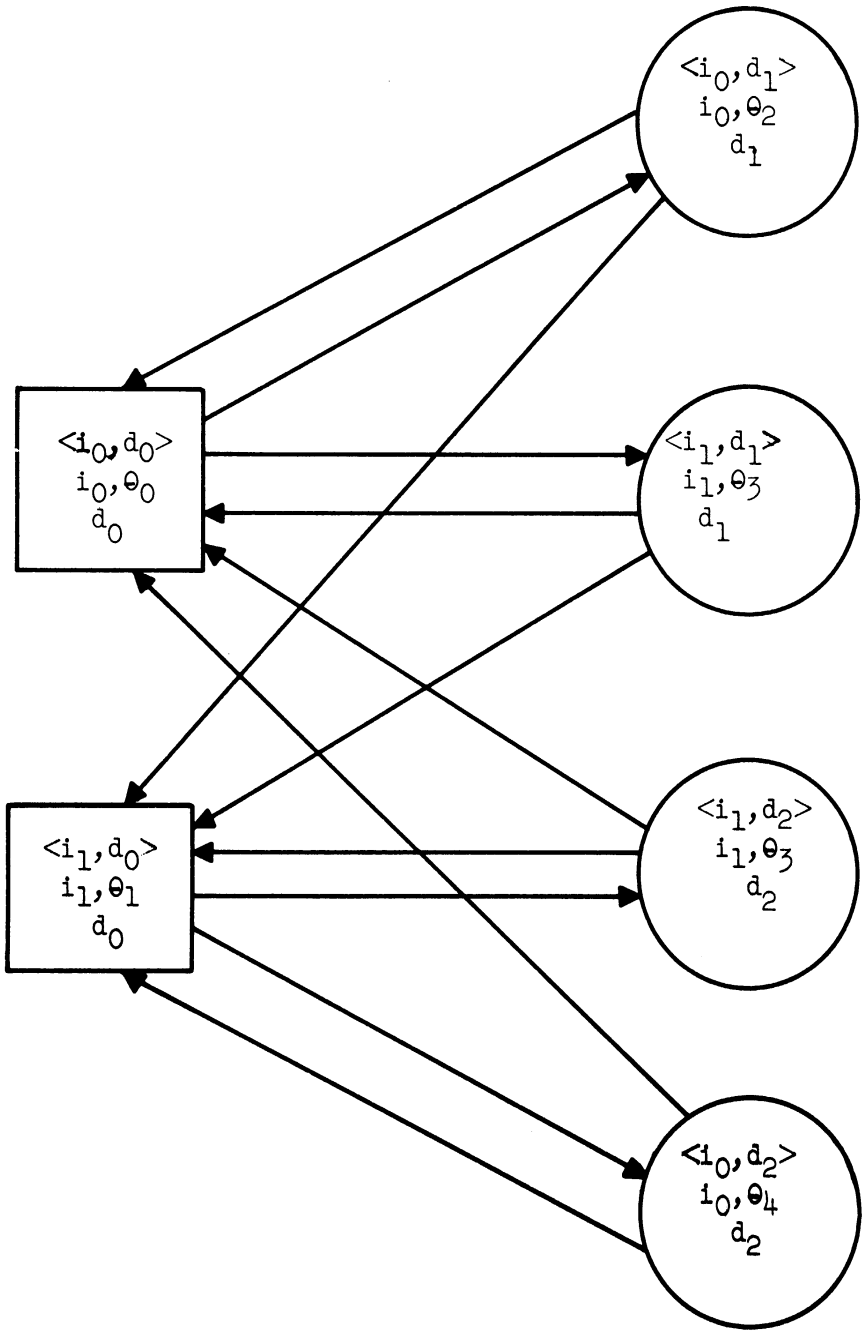
(b) Sequence generator neither solvable nor semi-deterministic

Figure 2.1-1



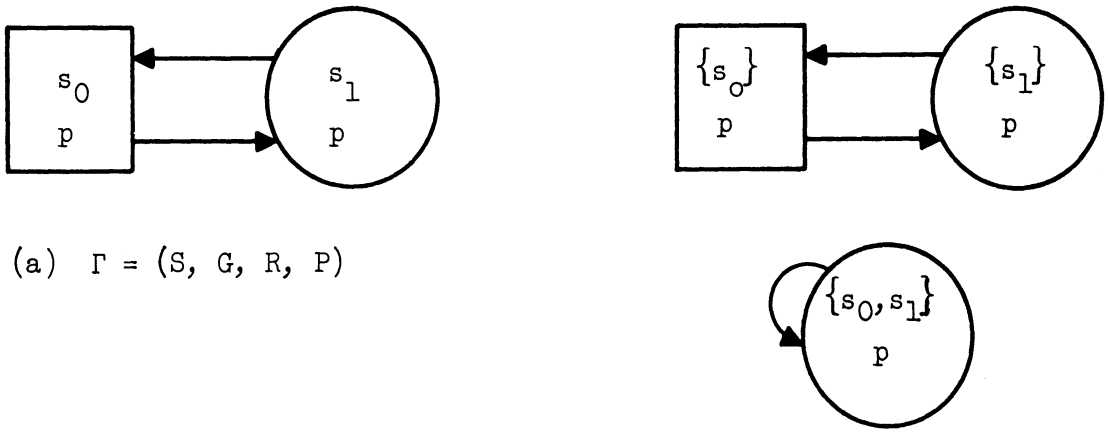
(a) Deterministic sequence generator $\Gamma = (S, G, R, I, \theta)$

Figure 2.1-2 (part)



(b) Internal state sequence generator
$\dot{\Gamma} = (\dot{S}, \dot{G}, \dot{R}, \dot{I}, \dot{\Theta}, \dot{D})$ corresponding to (a)

Figure 2.1-2 (part)

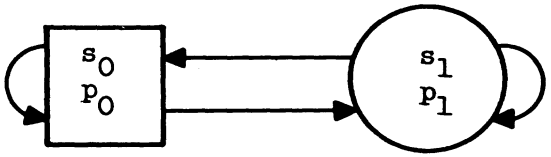


(a) $\Gamma = (S, G, R, P)$

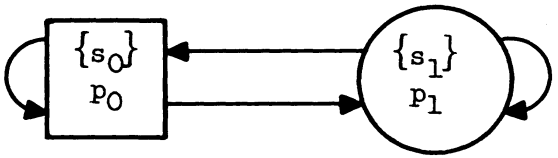
(b) Γ^*, the subset sequence generator of Γ

Figure 2.2-1

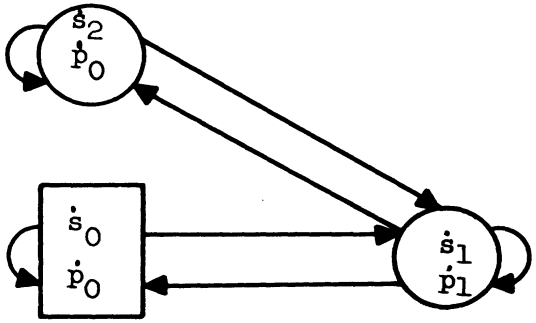
The construction of a subset sequence generator



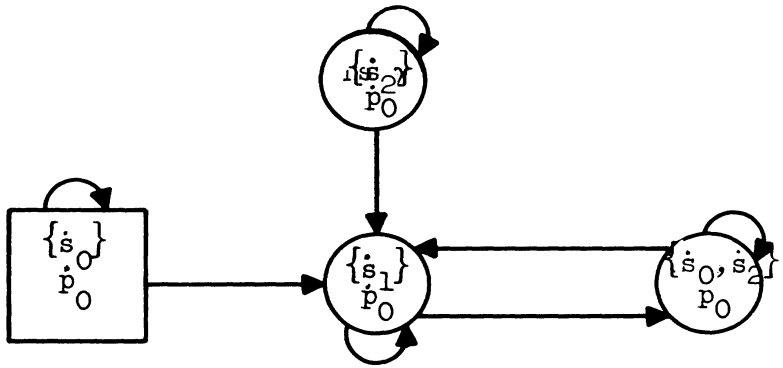
(a) Sequence generator
$\Gamma = (S, G, R, P)$.
Γ is semi-deterministic



(a') Γ^*, the subset sequence generator of Γ.
Γ^* is semi-deterministic



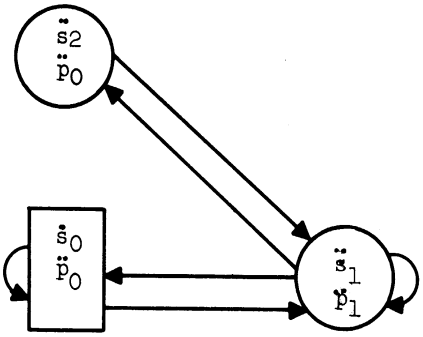
(b) Sequence generator
$\dot{\Gamma} = (\dot{S}, \dot{G}, \dot{R}, \dot{P})$.
$\dot{\Gamma}$ is not semi-deterministic



(b') $\dot{\Gamma}^*$, the subset sequence generator of $\dot{\Gamma}$.
$\dot{\Gamma}^*$ is semi-deterministic

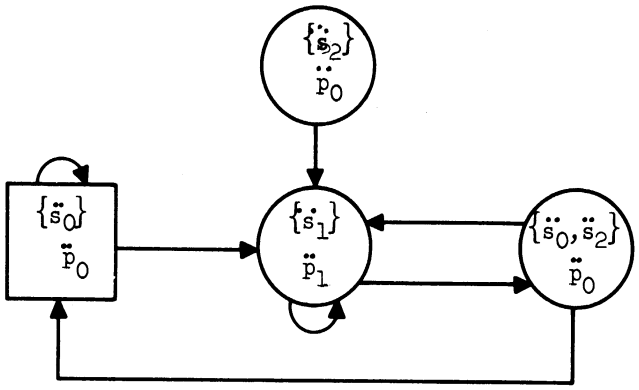
Figure 2.2-2 (part)

Examples which illustrate Lemma 2.2-1: For any sequence generator $\Gamma = (S, G, R, P)$ Γ^* is semi-deterministic.

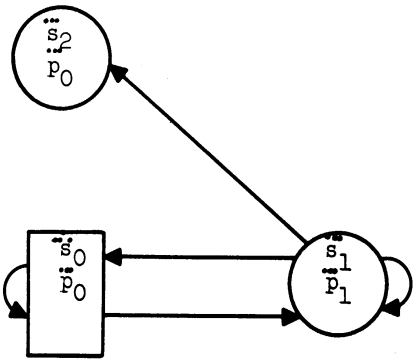


(c) $\ddot{\Gamma} = (\ddot{S}, \ddot{G}, \ddot{R}, \ddot{P})$

$\ddot{\Gamma}$ is not semi-deterministic

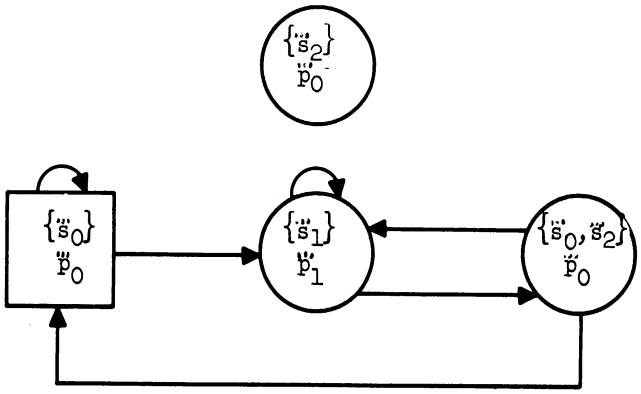


(c') $\ddot{\Gamma}$*, the subset sequence generator of $\ddot{\Gamma}$.
$\ddot{\Gamma}$* is semi-deterministic



(d) $\dddot{\Gamma} = (\dddot{S}, \dddot{G}, \dddot{R}, \dddot{P})$

$\dddot{\Gamma}$ is not semi-deterministic



(d') $\dddot{\Gamma}$*, the subset sequence generator of $\dddot{\Gamma}$.
$\dddot{\Gamma}$* is semi-deterministic

Figure 2.2-2 (continued)

3. Sequence Generators with Two Projections

3.1 Definitions

The results of the last section concern primarily one projection of a sequence generator. In the present section we will work mainly with two-projection sequence generators.

Definition: $\Gamma = (S, G, R, P, Q)$ is h-univalent ($h = 0, 1, 2, \ldots; \omega$) if for every two infinite Γ-sequences $[s_1](0,\omega)$, $[s_2](0,\omega)$ and any time t, if $P([s_1](0,t+h)) = P([s_2](0,t+h))$ then $Q(s_1(t)) = Q(s_2(t))$. (By definition, $t+\omega = \omega$.)

Note that h-univalence is essentially a property of a set of infinite sequences of pairs $\langle p,q \rangle$, and hence a property of $B^{\omega}(\Gamma)$, the infinite behavior of Γ. As a consequence the following lemma holds.

Lemma 3.1-1: Let $\Gamma = (S, G, R, P, Q)$ and $\dot{\Gamma} = (\dot{S}, \dot{G}, \dot{R}, \dot{P}, \dot{Q})$. If $B^{\omega}(\Gamma) = B^{\omega}(\dot{\Gamma})$ then Γ is h-univalent if and only if $\dot{\Gamma}$ is h-univalent. This lemma, together with Corollary 2.1-1 and Theorem 2.2-3, immediately yields Lemma 3.1-2: Let $\Gamma = (S, G, R, P, Q)$. The following three conditions are equivalent: (1) Γ is h-univalent, (2) Γ^* is h-univalent, (3) Γ^+ is h-univalent.

There is a close connection between zero-univalence and semi-determinism which is brought out by the following lemma.

Lemma 3.1-3: (a) Let $\Gamma = (S, G, R, P, Q)$ and $\Gamma = \Gamma^+$. Γ is 0-univalent if and only if for any two finite Γ-sequences $[s_1](0,t)$ and $[s_2](0,t)$, if $P([s_1](0,t)) = P([s_2](0,t))$ then $Q([s_1](0,t)) = Q([s_2](0,t))$. (b) Let $\Gamma = (S, G, R, P)$ and $\Gamma = \Gamma^+$. Γ is semi-deterministic if and only if for any two finite Γ-sequences, $[s_1](0,t)$ and $[s_2](0,t)$, if $P([s_1](0,t)) = P([s_2](0,t))$ then $[s_1](0,t) = [s_2](0,t)$.

Note that the sequence generator of part (a) of the lemma has two projections, while that of part (b) has one projection. Part (a) may be established by using Corollary 1.3-3 and the definition of univalence; part (b) follows from Corollary 1.3-3 and Lemma 2.1-4b. It follows from Lemma 3.1-3 that for any projection Q, if (S, G, R, P) is semi-deterministic then (S, G, R, P, Q) is 0-univalent. The converse is not in general true, but the following lemma asserts a connection between the 0-univalence of a sequence generator and the semi-determinism of a related sequence generator.

Lemma 3.1-4: Let $\Gamma = (S, G, R, P, Q)$ and let $\Gamma^{*+} = \dot{\Gamma} = (\dot{S}, \dot{G}, \dot{R}, \dot{P}, \dot{Q})$. Γ is zero-univalent if and only if $(\dot{S}, \dot{G}, \dot{R}, \dot{P})$ is semi-deterministic. It might seem that since $\dot{\Gamma}$ is the reduced form of the subset sequence generator of Γ, it would follow immediately by Lemma 2.2-1 that if Γ is 0-univalent then $(\dot{S}, \dot{G}, \dot{R}, \dot{P})$ is semi-deterministic. This is by no means the case. It can be shown from Lemma 2.2-1 by means of the definition of the subset sequence generator operation that $(\dot{S}, \dot{G}, \dot{R}, \dot{P}x\dot{Q})$ is semi-deterministic, while the conclusion of Lemma 3.1-4 is that $(\dot{S}, \dot{G}, \dot{R}, \dot{P})$, which is a different sequence generator, is semi-deterministic. For any projection $\dot{Q}$, if a sequence generator, (S, G, R, P) is semi-deterministic then $(\dot{S}, \dot{G}, \dot{R}, \dot{P}x\dot{Q})$ is semi-deterministic, but the converse is not in general true.

Proof of Lemma 3.1-4: ("Only if" part) We assume that Γ is 0-univalent and prove that $(\dot{S}, \dot{G}, \dot{R}, \dot{P})$ is semi-deterministic. (I) We will use three sequence generators in the proof besides Γ. These are: $\dot{\Gamma} = (\dot{S}, \dot{G}, \dot{R}, \dot{P}, \dot{Q})$; $\ddot{\Gamma} = (\dot{S}, \dot{G}, \dot{R}, \dot{P}x\dot{Q})$; and $\dddot{\Gamma} = (\dot{S}, \dot{G}, \dot{R}, \dot{P})$. We will first establish some results that will enable us to use Lemma 3.1-3a on $\dot{\Gamma}$ and Lemma 3.1-3b on $\ddot{\Gamma}$ and $\dddot{\Gamma}$. (A) By construction $\dot{\Gamma} = \dot{\Gamma}^{+}$ and by Lemma 3.1-2 $\dot{\Gamma}$ is semi-deterministic. (B) By construction $\ddot{\Gamma} = \ddot{\Gamma}^{+}$ and by Lemma 2.1-1 $\ddot{\Gamma}$ is semi-deterministic. (C) By construction $\dddot{\Gamma} = \dddot{\Gamma}^{+}$. Our task is to prove that $\dddot{\Gamma}$ is semi-deterministic. (II) Since $\dot{\Gamma}$, $\ddot{\Gamma}$, and $\dddot{\Gamma}$ have $\dot{S}$, $\dot{G}$, $\dot{R}$, in common, the sets of $\dot{\Gamma}$-sequences, $\ddot{\Gamma}$-sequences, and $\dddot{\Gamma}$-sequences are identical with one another. Consider now any two finite $\dot{\Gamma}$-sequences $[\dot{s}_1](0,t)$ and $[\dot{s}_2](0,t)$; these are also arbitrary $\ddot{\Gamma}$-sequences and arbitrary $\dddot{\Gamma}$-sequences. Using (IA) and applying Lemma 3.1-3a to $\dot{\Gamma}$ we obtain

(1) If $\dot{P}([\dot{s}_1](0,t)) = \dot{P}([\dot{s}_2](0,t))$ then $\dot{Q}([\dot{s}_1](0,t)) = \dot{Q}([\dot{s}_2](0,t))$.

Using (IB) and applying Lemma 3.1-3b to $\ddot{\Gamma}$ we obtain

(2) If $\dot{P}([\dot{s}_1](0,t)) = \dot{P}([\dot{s}_2](0,t))$ and $\dot{Q}([\dot{s}_1](0,t)) = \dot{Q}([\dot{s}_2](0,t))$ then $[\dot{s}_1](0,t) = [\dot{s}_2](0,t)$.

Combining (1) and (2) and noting that $[\dot{s}_1](0,t)$ and $[\dot{s}_2](0,t)$ are arbitrary $\ddot{\Gamma}$-sequences, we get

(3) For any two finite $\dddot{\Gamma}$-sequences $[\dot{s}_1](0,t)$ and $[\dot{s}_2](0,t)$, if $\dot{P}([\dot{s}_1](0,t)) = \dot{P}([\dot{s}_2](0,t))$ then $[\dot{s}_1](0,t) = [\dot{s}_2](0,t)$.

Using (3) and (IC) and applying Lemma 3.1-3b to $\dddot{\Gamma}$ we obtain

(4) $\dddot{\Gamma}$ is semi-deterministic,

which completes the proof of the "only if" part of the Lemma.

("If" part): We assume that $(\dot{S}, \dot{G}, \dot{R}, \dot{P})$ is semi-deterministic and prove that Γ is 0-univalent. Since every complete state of $\dot{\Gamma}$ is $\dot{\Gamma}$-accessible, by Lemma 2.1-4b we have that for every finite sequence of $\dot{P}$-states $[\dot{p}](0,t)$ there exists at most one $\dot{\Gamma}$-sequence $[\dot{s}](0,t)$ such that $\dot{P}\{[\dot{s}](0,t)\} = [\dot{p}](0,t)$. Hence for any two $\dot{\Gamma}$-sequences $[\dot{s}_1](0,\omega)$, $[\dot{s}_2](0,\omega)$ and any time t, if $\dot{P}([\dot{s}_1](0,t)) = \dot{P}([\dot{s}_2](0,t))$ then $\dot{s}_1(t) = \dot{s}_2(t)$. Since a projection is a (single-valued) function we have that if $\dot{P}([\dot{s}_1](0,t+0)) = \dot{P}([\dot{s}_2](0,t+0))$ then $\dot{Q}(\dot{s}_1(t)) = \dot{Q}(\dot{s}_2(t))$, so $\dot{\Gamma}$ is 0-univalent. $\dot{\Gamma} = \dot{\Gamma}^{*+}$, and by Lemma 3.1-2 Γ is 0-univalent. This completes the proof of Lemma 3.1-4.

We next apply this lemma to an example. Consider $\Gamma = (S, G, R, P, Q)$ of Figure 3.1-1a. Note that the complete states s_2 and s_4 have the same projections (p^0 and q^0) and stand in the same relation to state s_0. Thus the two Γ-sequences

$$s_0, s_4, s_0$$

$$s_0, s_2, s_0$$

have the same sequence of P-projections

$$p_0, p_0, p_0$$

and hence (S, G, R, P) is not semi-deterministic. These two Γ-sequences do have the same sequence of Q-Projections

$$q_0, q_2, q_0$$

and in fact Γ is 0-univalent. By Lemma 3.1-4 $(\dot{S}, \dot{G}, \dot{R}, \dot{P})$ of Figure 3.1-1b must be semi-deterministic. An examination of the states of $(\dot{S}, \dot{G}, \dot{R}, \dot{P})$ shows that it is deterministic, so <u>a fortiori</u> it is semi-deterministic. (The determinism of $(\dot{S}, \dot{G}, \dot{R}, \dot{P})$ will be discussed after Lemma 3.2-3 below.) Note that the main difference between Γ and Γ^{*+} in Figure 3.1-1 is that the two states s_2, s_4 of Γ have become a single state $\{s_2, s_4\}$ of Γ^{*+}.

Definition: $\Gamma = (S, G, R, P, Q)$ is uniquely solvable if (1) (S, G, R, P) is solvable and (2) (S, G, R, P, Q) is ω-univalent. We remarked earlier that h-univalence is essentially a property of the infinite behavior of a sequence generator, and this remark applies to unique solvability as well. Thus Γ is uniquely solvable if and only if for any infinite sequence of P-states $[p](0,\omega)$ there is exactly one sequence of Q-states $[q](0,\omega)$ such that the sequence $\langle p(0), q(0)\rangle, \langle p(1), q(1)\rangle, \ldots$ belongs to $B(\Gamma)$. To put the point in another way: a sequence generator $\Gamma = (S, G, R, P, Q)$ is uniquely solvable if and only if its behavior defines a single-valued function (transformation) from the set of all infinite sequences of P-states into the set of all infinite sequences of Q-states. Various consequences follow from this fact. The result of replacing "h-univalence" by "uniquely solvable" in Lemma 3.1-1 is also a Lemma. A similar remark holds for Lemma 3.1-2 except that Γ^+ may have fewer P-states (values of p) than Γ.

It was shown in Section 2.1 that well-formed nets and deterministic sequence generators are equivalent in a certain sense: for every w.f.n. there is a corresponding deterministic sequence generator and vice-versa. The w.f.n. gives the structure of an automaton while the associated deterministic sequence generator gives the corresponding complete state diagram. An analogous relation holds between the well-behaved nets of Burks and Wright, 1953, p. 1358 and uniquely solvable sequence generators. Consider any net and label all its non-input nodes as output nodes. The procedure of Section 1.2 will associate with this net a sequence generator which is uniquely solvable if and only if the original net is well-behaved.

3.2 The Displacement Operator and the ℓ-Shift Operation

We will first define a displacement operation D^k which applies to sets composed of finite sequences of pairs and/or ω-sequences of pairs. Roughly speaking D^k has the effect of leaving the first element of each pair where it is and displacing the second element of each pair k places to the right. Displacing the second element of the first pair k places to the right will leave k gaps, since the first pair is not preceded by any pair. It will be convenient always to fill these gaps with the same element; we will use the fixed state q_0 for this purpose.

Definition: Let the universe of discourse V consist of all finite sequences of pairs and all ω-sequences of pairs and let Λ be the null set. The operator D (without superscript) is defined to apply to any sequence of V as follows:

$$D(\langle x_0, y_0\rangle, \langle x_1, y_1\rangle, \langle x_2, y_2\rangle, \langle x_3, y_3\rangle, \ldots) =$$

$$(\langle x_0, \Lambda\rangle, \langle x_1, y_0\rangle, \langle x_2, y_1\rangle, \langle x_3, y_2\rangle, \ldots)$$

$$D(\langle x_0, y_0\rangle, \langle x_1, y_1\rangle, \ldots, \langle x_{n-1}, y_{n-1}\rangle, \langle x_n, y_n\rangle) =$$

$$(\langle x_0, \Lambda\rangle, \langle x_1, y_0\rangle, \ldots, \langle x_{n-1}, y_{n-2}\rangle, \langle x_n, y_{n-1}\rangle).$$

The operator D is extended to apply to an arbitrary set α of V by

$$D(\alpha) = \{v \mid (\exists u)\ [u \in \alpha \ \&\ v = D(u)]\},$$

where v and u range over elements of V. Finally, we define D^k, k = 0,1,2,..., to apply to an arbitrary set α of V by the induction

$$D^0(\alpha) = \alpha$$

$$D^{i+1}(\alpha) = D(D^i(\alpha))$$

D^ℓ is called the <u>displacement operator</u>.

We next define a shifting operation which may be applied to an arbitrary sequence generator $\Gamma = (S, G, R, P, Q)$ to produce the ℓ-shifted sequence generator $\Gamma^\ell = \dot{\Gamma} = (\dot{S}, \dot{G}, \dot{R}, \dot{P}, \dot{Q})$. The effect of this operation is to displace the behavior of Γ, so that the behavior of Γ^ℓ, i.e., $B(\Gamma^\ell)$, equals the displaced behavior of Γ, i.e., $D^\ell[B(\Gamma)]$, as is shown in Lemma 3.2-1 below. To help make clear the definition of Γ^ℓ, we will make some remarks about Γ^1. Extend Q to apply to Λ, so that $Q(\Lambda) = \Lambda$. The generators of Γ^2 are the pairs $\langle \Lambda, s\rangle$, where s belongs to G. Suppose

$$s_0, s_1, s_2, s_3, s_4$$

is a Γ-sequence with the resulting behavior element

$$\langle p_0, q_0\rangle, \langle p_1, q_1\rangle, \langle p_2, q_2\rangle, \langle p_3, q_3\rangle, \langle p_4, q_4\rangle.$$

Then

$$\langle \Lambda, s_0\rangle, \langle s_0, s_1\rangle, \langle s_1, s_2\rangle, \langle s_2, s_3\rangle, \langle s_3, s_4\rangle$$

is the corresponding Γ^1-sequence with the resulting behavior element

$$\langle p_0, \Lambda\rangle, \langle p_1, q_0\rangle, \langle p_2, q_1\rangle, \langle p_3, q_3\rangle, \langle p_4, q_3\rangle.$$

Γ^ℓ may be obtained by shifting Γ ℓ times in this way.

Definition: The unit-shift operation, denoted by "$\langle\rangle$", applies to any sequence generator $\Gamma = (S, G, R, P, Q)$ and produces a sequence generator $\Gamma^{\langle\rangle} = \dot{\Gamma} = (\dot{S}, \dot{G}, \dot{R}, \dot{P}, \dot{Q})$ defined as follows.

(1) The elements of $\dot{G}$ are all the pairs $\langle\Lambda, s\rangle$ where s belongs to G:

$$\langle\Lambda, s\rangle \in \dot{G} \equiv s \in G$$

(2) The elements of $\dot{S}$ are all the pairs $\langle s_1, s_2\rangle$ which either belong to $\dot{G}$ or are connected by the direct transition relation R:

$$\dot{S} = \{\langle s_1, s_2\rangle \mid \langle s_1, s_2\rangle \in \dot{G} \vee R(s_1, s_2)\}$$

(3) Two complete states $\langle s_1, s_2\rangle$ and $\langle s_3, s_4\rangle$ of $\dot{S}$ stand in the direct transition relation $\dot{R}$ if and only if $s_2 = s_3$:

$$\dot{R}(\langle s_1, s_2\rangle, \langle s_3, s_4\rangle) \equiv [\langle s_1, s_2\rangle, \langle s_3, s_4\rangle \in \dot{S} \;\&\; s_2 = s_3].$$

(4) The $\dot{P}$-projection of a complete state $\langle s_1, s_2\rangle$ of $\dot{S}$ is the P-projection of its second element s_2:

$$\dot{P}(\langle s_1, s_2\rangle) = P(s_2), \text{ where } \langle s_1, s_2\rangle \in \dot{S}.$$

(5) Extend Q to apply to Λ, stipulating that $Q(\Lambda) = \Lambda$. The $\dot{Q}$-projection of the complete state $\langle s_1, s_2\rangle$ of $\dot{S}$ is the Q-projection of it, first element:

$$\dot{Q}(\langle s_1, s_2\rangle) = Q(s_1), \text{ where } \langle s_1, s_2\rangle \in \dot{S}.$$

The ℓ-shift operation, denoted by "ℓ", applies to any sequence generator $\Gamma = (S, G, R, P, Q)$ and produces a sequence generator Γ^{ℓ}; it is defined in terms of the unit-shift operation by means of an induction:

$$\Gamma^{0} = \Gamma$$

$$\Gamma^{\ell+1} = (\Gamma^{\ell})^{\langle\rangle}$$

The ℓ-shift construction is illustrated in Figure 3.2-1. Part (a) shows a sequence generator Γ with two projections, while part (b) shows Γ^{1}, the result of shifting Γ one unit of time. It should be noted that the generator $\langle\Lambda, s_0\rangle$ can only occur as the first state of a $\dot{\Gamma}$-sequence; an examination of the definition of the ℓ-shift operation shows that the generators of any Γ^{ℓ} can only occur as the first states of Γ^{ℓ}-sequences.

(We are assuming throughout that Λ is not an element of S; if it is, S should be redefined so it is not.) Note also that both Γ and Γ^1 are in reduced form; this is a special case of the general fact that if Γ is in reduced form then Γ^h is in reduced form. We will discuss next the behaviors of Γ and Γ^1. The behavior element [and the element of $B(\Gamma)$]

$$(1)\quad \langle p_0, q_0\rangle, \langle p_0, q_0\rangle, \langle p_1, q_1\rangle, \langle p_0, q_0\rangle$$

is derived from the Γ-sequence

$$s_0, s_0, s_1, s_1, s_0$$

The corresponding Γ^1-sequence is

$$\langle \Lambda, s_0\rangle, \langle s_0, s_0\rangle, \langle s_0, s_1\rangle, \langle s_1, s_1\rangle, \langle s_1, s_0\rangle$$

which gives rise to the behavior element [an element of $B(\Gamma^1)$]

$$(2)\quad \langle p_0, \Lambda\rangle, \langle p_0, q_0\rangle, \langle p_1, q_0\rangle, \langle p_1, q_1\rangle, \langle p_0, q_1\rangle.$$

Note that this last sequence (2) is the result of displacing sequence (1) by one unit. This is an example of the general fact that $B(\Gamma^1) = D^1[B(\Gamma)]$, which is a special case of the following lemma.

Lemma 3.2-1: Let $\Gamma = (S, G, R, P, Q)$. Then

$$(a)\quad D^\ell[B(\Gamma)] = B(\Gamma^\ell)$$

$$(b)\quad D^\ell[B^\omega(\Gamma)] = B^\omega(\Gamma^\ell)$$

Proof: (IA) We prove first that $D[B^\omega(\Gamma)] = B^\omega(\Gamma^{\diamond})$. Let $\Gamma^{\diamond} = \dot{\Gamma} = (\dot{S},\dot{G},\dot{R},\dot{P},\dot{Q})$. It follows from the definition of the unit shift operation that there is a one-one correspondence between the set of infinite Γ-sequences and the set of infinite $\dot{\Gamma}$-sequences with corresponding sequences being of the form

$$(1)\quad s_0, s_1, s_2, s_3, \ldots$$

$$(2)\quad \langle \Lambda, s_0\rangle, \langle s_0, s_1\rangle, \langle s_1, s_2\rangle, \langle s_2, s_3\rangle, \ldots$$

When P x Q is applied to (1) we get

$$(3)\quad \langle p_0, q_0\rangle, \langle p_1, q_1\rangle, \langle p_2, q_2\rangle, \langle p_3, q_3\rangle, \ldots$$

as an element of $B^{\omega}(\Gamma)$ and when $\dot{P} \times \dot{Q}$ is applied to (2) we get

(4) $\langle p_0, \Lambda\rangle, \langle p_1, q_0\rangle, \langle p_2, q_1\rangle, \langle p_3, q_2\rangle, \ldots$

as an element of $B^{\omega}(\dot{\Gamma})$. By definition of D,

$$D[(3)] = (4).$$

Hence $D[B^{\omega}(\Gamma)] = B^{\omega}(\Gamma^{\diamond})$.

(IB) Applying mathematical induction to result (IA) we get

$$D^{\ell}[B^{\omega}(\Gamma)] = B^{\omega}(\Gamma^{\ell})].$$

(IIA) An argument similar to that of (IA) may be given for finite Γ-sequences and finite $\Gamma^{\diamond}$-sequences. When the result is combined with result (IA) we get

$$D[B(\Gamma)] = B(\Gamma^{\diamond}).$$

(IIB) Applying mathematical induction to (IIA) we get

$$D^{\ell}[B(\Gamma)] = B(\Gamma^{\ell})$$

Corollary 3.2-2: Let $\Gamma = (S, G, R, P, Q)$ and $\Gamma^{\ell} = \dot{\Gamma} = (\dot{S}, \dot{G}, \dot{R}, \dot{P}, \dot{Q})$. [$\Gamma$ is $(\ell + h)$-univalent] $\{(S, G, R, P)$ is solvable$\}$ (Γ is uniquely solvable) if and only if [Γ^{ℓ} is h-univalent] $\{(\dot{S}, \dot{G}, \dot{R}, \dot{P})$ is solvable$\}$ (Γ^{ℓ} is uniquely solvable).

This corollary is illustrated by Figure 3.2-2. Consider Γ of this figure. It was shown in Section 2.1 (in the paragraph preceding Lemma 2.1-4) that (S, G, R, P) is solvable. Also, Γ is unit-univalent. To see this observe that every immediate successor (by the direct transition relation R) of a given complete state s has the same P-projection; e.g., $R(s_0) = \{s_0, s_1\}$ and $P(s_0) = P(s_1) = P_0$. Since (S, G, R, P) is solvable and Γ is unit-univalent, Γ is uniquely solvable. Turn now to Γ^1, the unit-shifted sequence generator of Γ. Since (S, G, R, P) is solvable and Γ is unit-univalent and uniquely solvable, by Corollary 3.2-2, Γ^1 (less its last projection) must be solvable, and Γ^1 must be zero-univalent and also uniquely solvable.

Lemma 3.2-3: Let $\Gamma = (S,G,R,P,Q)$ satisfy the conditions (1) Γ is h-univalent for some finite h (2) (S,G,R,P) is solvable. Let $\Gamma^{h*+} = \dot{\Gamma} = (\dot{S},\dot{G},\dot{R},\dot{P},\dot{Q})$. Then (a) $(\dot{S}, \dot{G}, \dot{R}, \dot{P})$ is deterministic (b) $D^{h}[B^{\omega}(\Gamma)] = B^{\omega}(\dot{\Gamma})$.

Proof: (IA) We will prove first that $(\dot{S}, \dot{G}, \dot{R}, \dot{P})$ is solvable. It is given that (S, G, R, P) is solvable. By Corollary 3.2-2, Lemma 2.2-4, and Lemma 2.1-4a, $(\dot{S}, \dot{G}, \dot{R}, \dot{P})$ is solvable. (IB) We prove next that $(\dot{S}, \dot{G}, \dot{R}, \dot{P})$ is semi-deterministic. It is given that Γ is h-univalent. By Corollary 3.2-2, Γ^h is 0-univalent. By Lemma 3.1-4, $(\dot{S}, \dot{G}, \dot{R}, \dot{P})$ is semi-deterministic. (IC) It follows from Lemma 2.1-4b and the definition of solvable that if every complete state of any sequence generator $\Gamma = (S, G, R, P)$ is Γ-accessible, than Γ is deterministic if and only if Γ is both semi-deterministic and solvable. Applying this principle to $(\dot{S}, \dot{G}, \dot{R}, \dot{P})$ and using (IA) and (IB) we conclude that $(\dot{S}, \dot{G}, \dot{R}, \dot{P})$ is deterministic. This proves part (a) of the lemma.

(II) By Lemma 3.2-1b

$$(I) \quad D^h[B^\omega(\Gamma)] = B^\omega(\Gamma^h).$$

By Theorem 2.2-3 $B(\Gamma^h) = B(\Gamma^{h*})$ and hence

$$(2) \quad B^\omega(\Gamma^h) = B^\omega(\Gamma^{h*}).$$

But by Corollary 2.1-1

$$(3) \quad B^\omega(\Gamma^{h*}) = B^\omega(\Gamma^{h*+}).$$

Combining (1), (2), and (3) gives part (b) of Lemma 3.2-3 and completes the proof of the present lemma.

We may apply this lemma to Figure 3.1-1. As noted in Section 3.1 (S,G,R,P) is not semi-deterministic but (S, G, R, P,Q) is 0-univalent. It is easy to see that (S, G, R, P) is solvable. Applying Lemma 3.2-3 with h = 0 we conclude that $(\dot{S},\dot{G},\dot{R},\dot{P})$ is deterministic and that $B^\omega(\Gamma) = B^\omega(\dot{\Gamma})$. These two facts may be confirmed by inspection of Figure 3.1-b.

Actually $B(\Gamma) = B(\dot{\Gamma})$ in Figure 3.1-1, i.e., the finite behaviors of Γ and $\dot{\Gamma}$ are equal as well as the infinite behaviors. There is a variant of Lemma 3.2-3 which covers this point. Since our main interest in the present section is in infinite behavior we will merely state this result without proof. Let $\Gamma = (S,G,R,P,Q)$ be h-univalent and (S,G,R,P) solvable; Then Γ^{h*} less its Q-projection is deterministic, $D^h[B(\Gamma)] = B(\Gamma^{h*})$, and if Γ is in reduced form then $B(\Gamma^{h*}) = B(\Gamma^{h*+})$.

Figure 3.2-3 also illustrates Lemma 3.2-3. We begin with $\Gamma = (S, G, R, P, Q)$, where Γ is unit-univalent and (S, G, R, P) is solvable. Lemma 3.2-3 tells us that Γ^{1*+} (less its last projection) is deterministic, and also that $D^1[B^\omega(\Gamma)] = B^\omega(\Gamma^{2x+})$. ($\Gamma^1$ is shown in Figure 3.2-2b; it has 12 complete states. Γ^{1*} has 28 states, but only 6 of these are Γ^{1*}-admissible, so Γ^{1*+} has only 6 states.)

3.3 Time-Shift Theorem

We will prove now a lemma which is used in proving one of our main theorems (the Time-shift theorem) and in validating a procedure for h-univalence.

Lemma 3.3-1 (Fixed Bound Lemma): Let $\Gamma = (S,G,R,P,Q)$ be a sequence generator with k Γ-admissible complete states. Then Γ is ω-univalent if and only if it is k^2-univalent.

Proof: The proof is one direction is obvious. To prove that if Γ is ω-univalent it is k^2-univalent we consider any two Γ-sequences $[s_1](0,\omega)$, $[s_2](0,\omega)$ and any time t such that $P([s_1](0,t+k^2)) = P([s_2](0,t+k^2))$. Since there are k^2 distinct pairs of complete states, there are two times t_1, t_2 such that $t \leq t_1 < t_2 \leq t + k^2$, $s_1(t_1) = s_1(t_2)$, and $s_2(t_1) = s_2(t_2)$.

Form the sequences

$$[s_3](0,\omega) = [s_1](0,t_2-1),\ [s_1](t_1,t_2-1),\ [s_1](t_1,\ t_2-1),\ldots$$

$$[s_4](0,\omega) = [s_2](0,t_2-1),\ [s_2](t_1,t_2-1),\ [s_2](t_1,t_2-1)\ldots$$

These are both Γ-sequences since they are composed of segments of Γ-sequences linked by the direct transition relation. Since $P([s_1](0,t+k^2)) = P)[s_2](0,t+k^2))$ we have by construction $P([s_3](0,\omega)) = P([s_4)(0,\omega))$. Because Γ is ω-univalent, $Q([s_3](0,\omega)) = Q([s_4](0,\omega))$. Then by construction $Q(s_1(t)) = Q(s_3(t))$ and $Q(s_2(t)) = Q(s_4(t))$, and so $Q(s_1(t)) = Q(s_2(t))$. Hence Γ is k^2-univalent.

Consider a sequence generator $\Gamma=(S,G,R,P,Q)$. If (S,G,R,P) is deterministic then (S,G,R,P,Q) is uniquely solvable, but the converse does not in general hold (see Figure 3.1-1a). We noted earlier (Section 3.1) that unique solvability is essentially a property of the infinite behavior of a sequence generator. This suggests the question: What is the relation of the behaviors of uniquely solvable sequence generators to the behaviors of deterministic ones? This question is answered by the following theorem,

which shows that for every uniquely solvable sequence generator there is a deterministic sequence generator whose infinite behavior is a displacement of the infinite behavior of the given sequence generator. In Section 4 we will introduce a concept of "computation". Using this concept the result may be expressed: the behavior of every uniquely solvable sequence generator can be computed by a finite automaton.

Theorem 3.3-2 (Time-shift Theorem): Let $\Gamma = (S,G,R,P,Q)$ be a uniquely solvable sequence generator with k Γ-admissible complete states and let $\Gamma^{k^2*+} = \dot{\Gamma} = (\dot{S},\dot{G},\dot{R},\dot{P},\dot{Q})$. Then

(a) $(\dot{S},\dot{G},\dot{R},\dot{P})$ is deterministic

(b) $D^{k^2}[B^{\omega}(\Gamma)] = B^{\omega}(\dot{\Gamma})$.

Proof: This follows immediately from the definition of uniquely solvable (Section 3.1), Lemma 3.2-3, and the Fixed Bound Lemma (3.3-1).

Consider the Time-shift theorem in relation to Γ of Figure 3.2-2a and the derived Γ^{1*+} of Figure 3.2-3. Γ is uniquely solvable and has 4 Γ-admissible complete states. Then the time-shift theorem tells us that Γ^{16*+}, less its last projection, is deterministic. This is clearly so, for Γ^{1*+}, less its last projection, is deterministic, and further applications of the ℓ-shift operation will obviously not destroy this property.

We pause to note an analogue of the Time-shift Theorem in which the shifting takes place in the opposite direction. The displacement operator D^{ℓ} was defined to produce a right-shift of the Q-projections of a Γ-sequence; that is, it shifts the Q-projections ℓ steps later in time, leaving the P-projections as they were. One could easily extend this operator to cover shifts in the opposite direction (i.e., with the Q-projections moved earlier in time); this could be symbolized by using the same operator D^{ℓ}, allowing negative as well as positive integer values for ℓ. Similarly the ℓ-shift operator can be extended to produce shifts of the Q-projections to the left; again, we can use the same symbolism Γ^{ℓ} and signify left-shifts by negative values of ℓ. We then get the following partial analogue to the time-shift theorem. Let $\Gamma = (S,G,R,P,Q)$ be a sequence generator, with (S,G,R,P) deterministic. Let $\dot{\Gamma} = \Gamma^{\ell}$, where ℓ is negative. Then $\dot{\Gamma}$ is uniquely solvable, and $D^{\ell}[B^{\omega}(\Gamma)] = B^{\omega}(\dot{\Gamma})$. Combining this with the Time-shift Theorem we obtain the following result: the set of infinite behaviors of uniquely solvable sequence generators is exactly the set of displaced infinite behaviors of deterministic sequence generators.

It is not obvious from the definition that the class of h-univalent sequence generators is decidable. However, this is in fact the case, as we will now show.

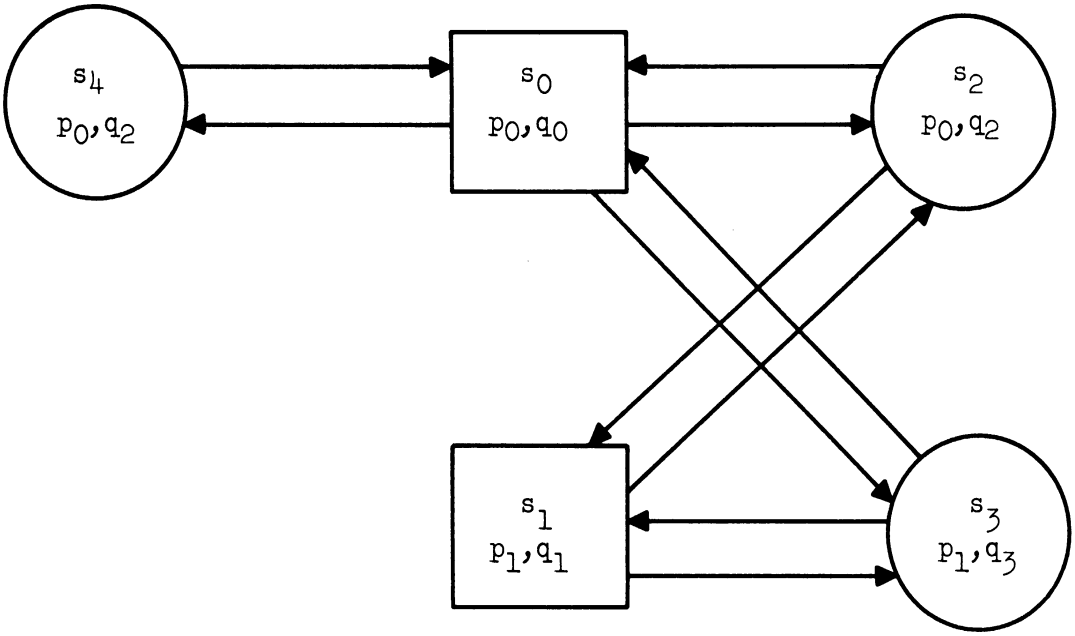
h-univalence Procedure (where h is any non-negative integer or ω): Let $\Gamma = (S,G,R,P,Q)$ be the given sequence generator. Find k, the number of admissible complete states, by the Reduced Form Algorithm. Let $\ell = \min(h,k^2)$. Form $\Gamma^{\ell *+} = \dot{\Gamma} = (\dot{S},\dot{G},\dot{R},\dot{P},\dot{Q})$. Answer "yes" or "no" as $(\dot{S},\dot{G},\dot{R},\dot{P})$ is semi-deterministic or not.

Theorem 3.3-3: The h-univalence procedure is a decision procedure for the class of h-univalent sequence generators.

Proof: We will use the notation of the algorithm. By the Fixed Bound Lemma Γ is h-univalent if and only if Γ is ℓ-univalent. By Corollary 3.2-2 Γ is ℓ-univalent if and only if Γ^{ℓ} is 0-univalent. By Lemma 3.1-4 Γ^{ℓ} is 0-univalent if and only if $(\dot{S},\dot{G},\dot{R},\dot{P})$ is semi-deterministic. As noted in Section 2.1, it is obvious from the definition of semi-determinism that there is a decision procedure for the class of semi-deterministic sequence generators. This completes the proof of the theorem.

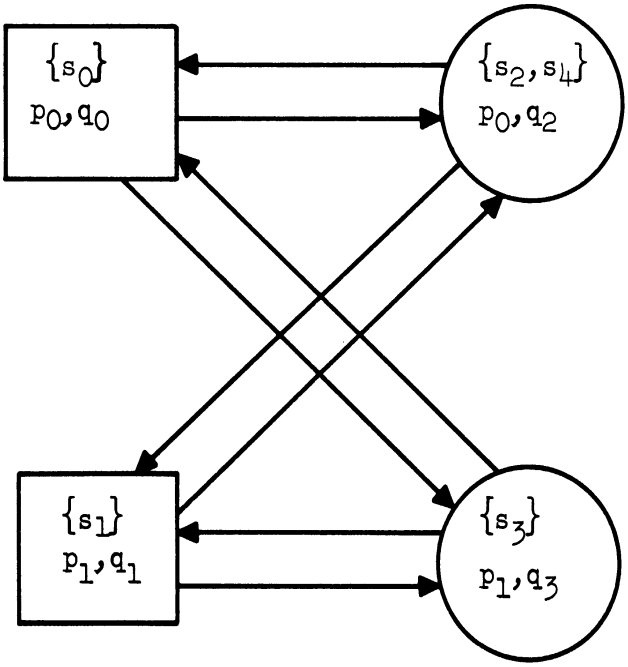
It can be shown that the following is a characterization of h-univalence. Let $\Gamma = (S,G,R,P,Q)$, k the number of Γ-admissible complete states, and $\ell = \min(h,k^2)$. Then Γ is h-univalent, if and only if, for any two Γ-sequences $[s_1](0,\omega)$ and $[s_2](0,\omega)$ and any time $t \leq k^2$, if $P([s_1](0,t+\ell)) = P([s_2](0,t+\ell))$ then $Q(s_1(t)) = Q(s_2(t))$. This characterization can be made the basis of a decision procedure for h-univalence which is more efficient than the one we have given.

Since unique solvability is defined in terms of solvability and ω-univalence (Section 3.1), by combining the ω-univalence procedure with the decision procedure for solvability of Theorem 2.3-2, we obtain a decision procedure for unique solvability.



(a) $\Gamma = (S, G, R, P, Q)$.

Γ is zero-univalent and uniquely solvable, but (S,G,R,P) is not semi-deterministic.

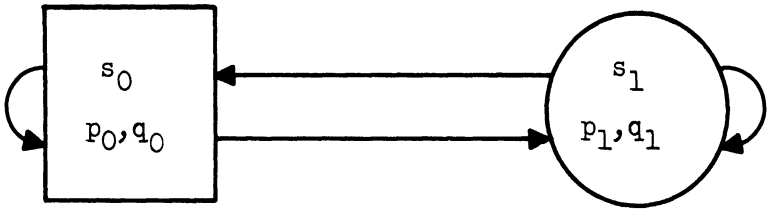


(b) $\Gamma^{*+} = \dot{\Gamma} = (\dot{S}, \dot{G}, \dot{R}, \dot{P}, \dot{Q})$.

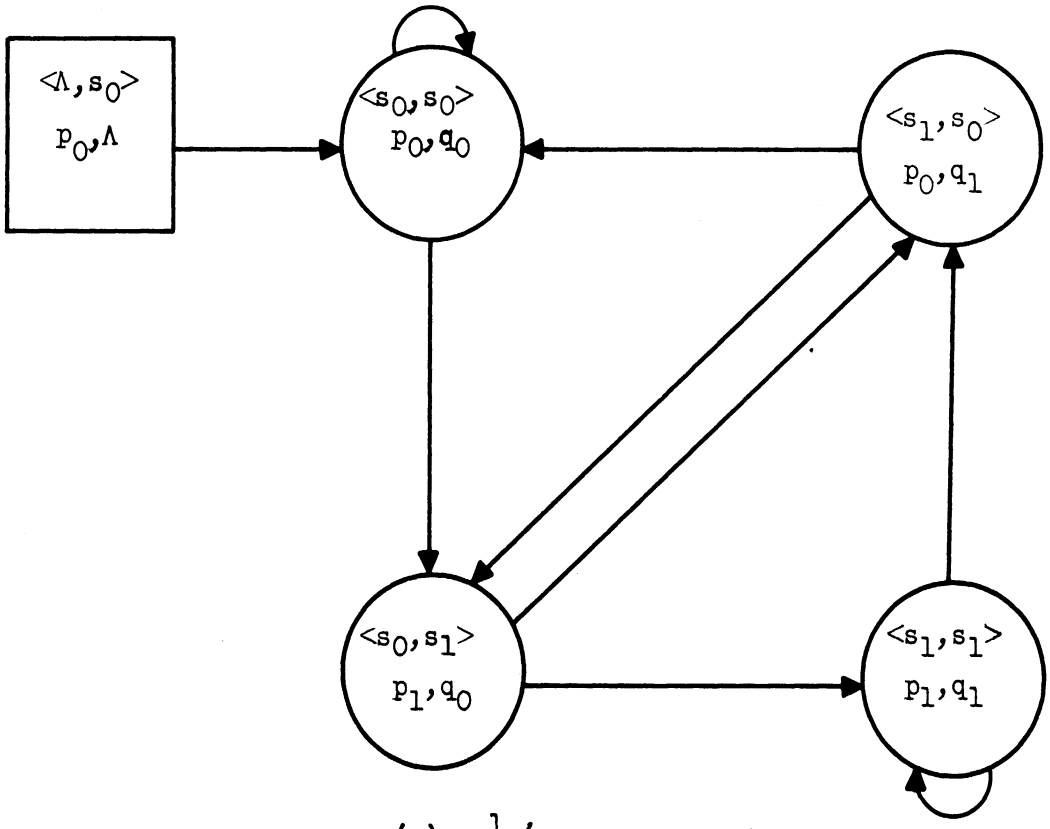
$(\dot{S}, \dot{G}, \dot{R}, \dot{P})$ is deterministic, and a fortiori semi-deterministic

Figure 3.1-1

Illustration of Lemma 3.1-4: Γ is zero-univalent if and only if $(\dot{S}, \dot{G}, \dot{R}, \dot{P})$ is semi-deterministic.



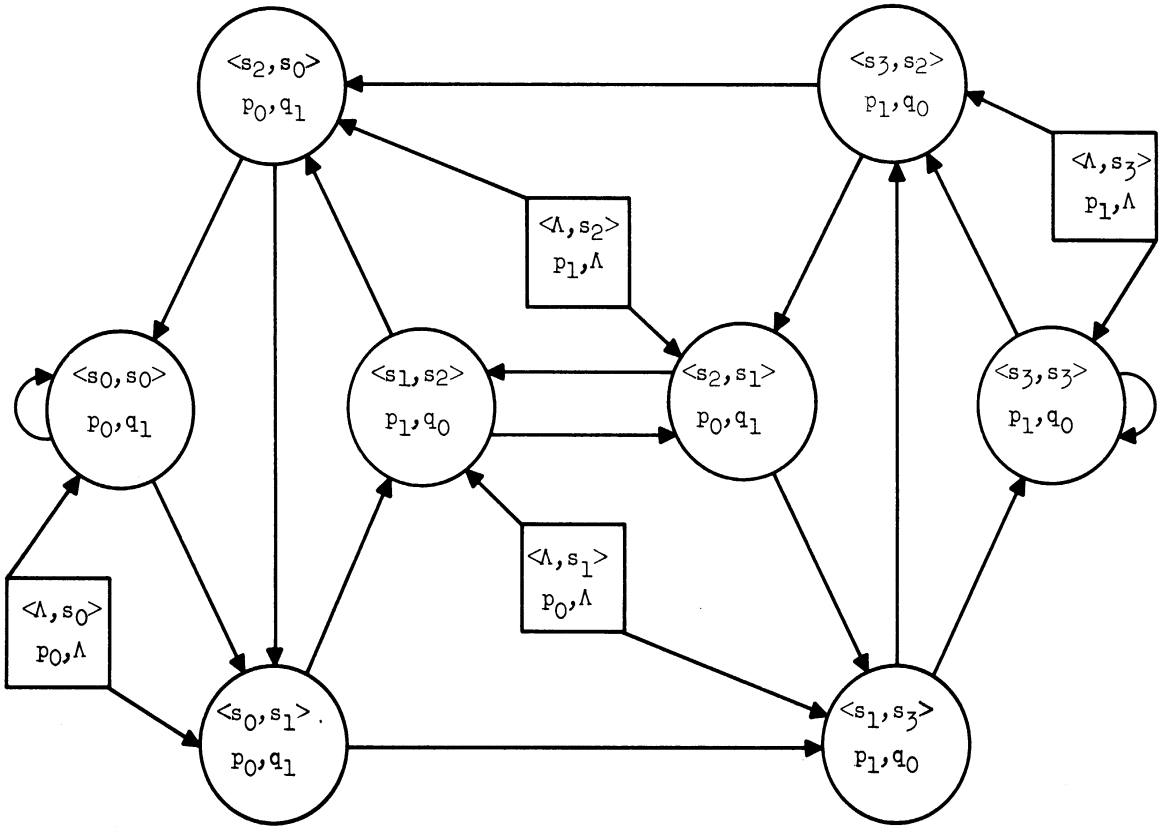
(a) $\Gamma = (S, G, R, P, Q)$



(b) Γ^1 (1 unit-shifted sequence generator of Γ)

Figure 3.2-1

Illustration of the ℓ-shift construction, for $\ell = 1$. In accordance with Lemma 3.2-1, $D'[B(\Gamma)] = B(\Gamma')$.



(b) Γ^1, the unit-shifted sequence generator of Γ.
Γ^1, less its last projection, is solvable,
but not deterministic.
Γ^1, is zero-univalent and uniquely solvable.

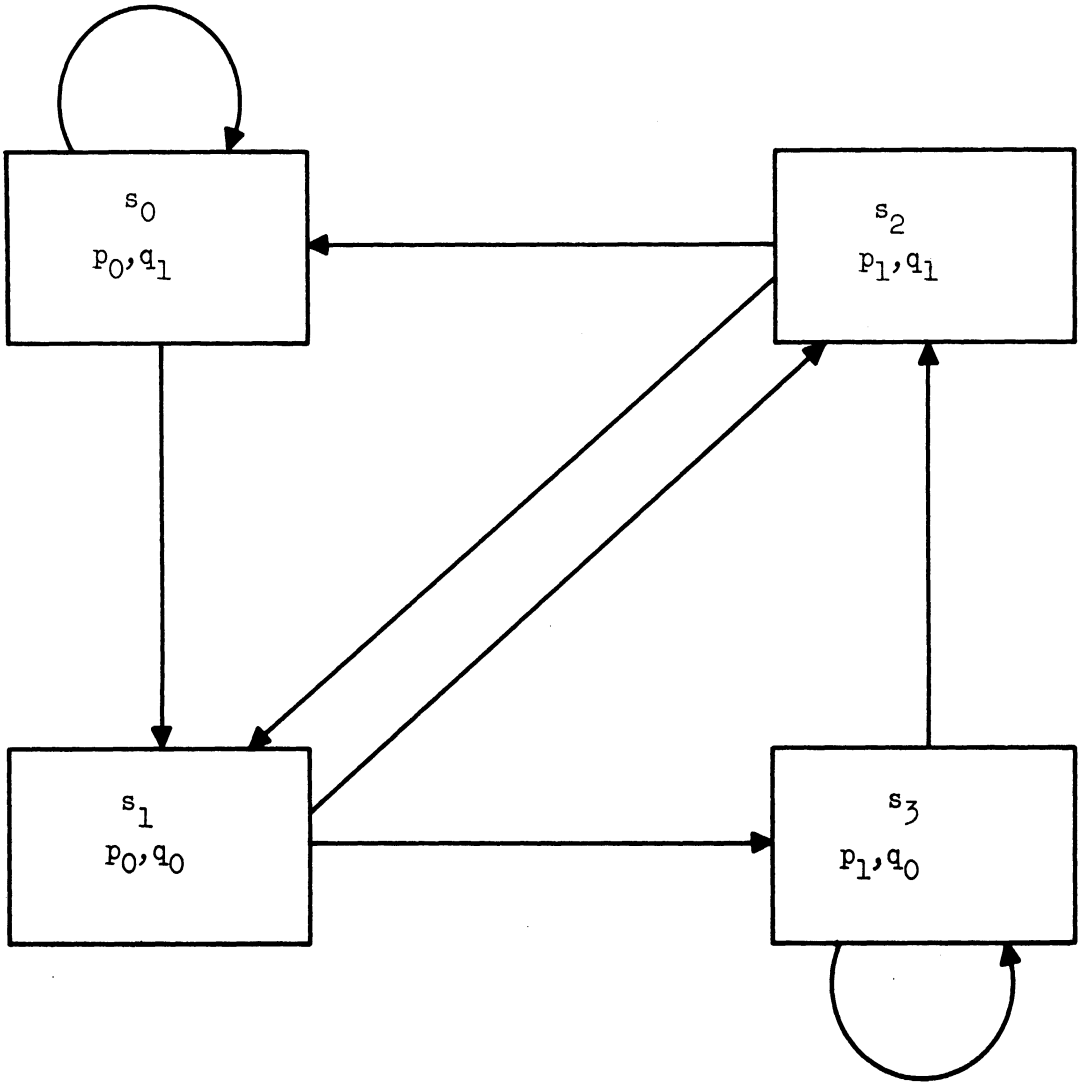
Figure 3.2-2

Illustration of Corollary 3.2-2

[Γ is 1-univalent] {Γ, less its last projection, is solvable} (Γ is uniquely solvable)

if and only if

[Γ^1 is 0-univalent] {Γ^1, less its last projection, is solvable} (Γ^1 is uniquely solvable)



(a) $\Gamma = (S, G, R, P, Q)$

(S, G, R, P) is solvable but <u>not</u> deterministic.

Γ is unit-univalent and uniquely solvable.

Figure 3.2-2a

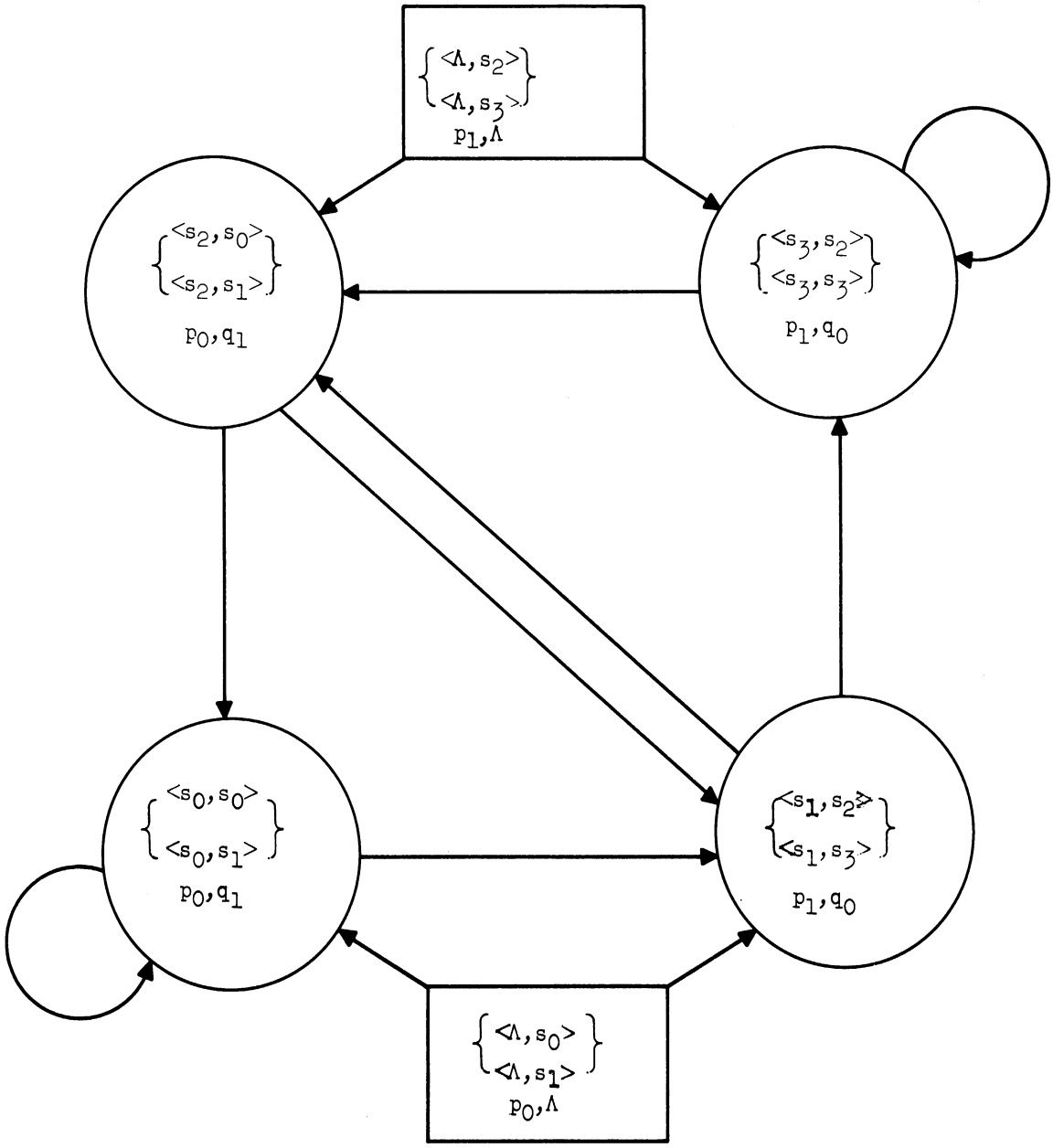


Figure 3.2-3

Γ^{1*+}, where Γ is Figure 3.2-2a
Γ and Γ^1 (less their last projections) are not deterministic, but
Γ^{1*+} (less its last projection) is deterministic
$D^1[B^\omega(\Gamma)] = B^\omega(\Gamma^{1*+})$
This illustrates Lemma 3.2-3.

BIBLIOGRAPHY

Aufenkamp, D. O. and Hohn, F. E., "Analysis of Sequential Machines," Institute of Radio Engineers Transactions on Electronic Computers EC-6 (1957), 276-285.

Burks, A. W., "The Logic of Fixed and Growing Automata," in Proceedings of an International Symposium on the Theory of Switching, 2-5 April, 1957, Cambridge: Harvard University Press, 1959, Part I, pp. 147-188.

Burks, A. W. and Wang, H. "The Logic of Automata," Journal of the Association for Computing Machinery 4 (1957), 193-218, 279-297.

Burks, A. W. and Wright, J. B. "Theory of Logical Nets," Proceedings of the Institute of Radio Engineers 41 (1953), 1357-1365.

Chomsky, N. and Miller, G. A. "Finite State Languages," Information and Control 1 (1958), 91-112.

Church, A., The Journal of Symbolic Logic 20 (1955), 286-287.

Copi, I. M., Elgot, C. C. and Wright, J. B., "Realization of Events by Logical Nets," Journal of the Association for Computing Machines 5 (1958), 181-196.

Fitch, F. B., "Representation of Sequential Circuits in Combinatory Logic," Philosophy of Science 25 (1958), 263-279.

Harary, F. and Paper, H. H. "Toward a General Calculus of Phonemic Distribution," Language 33 (1957), 143-169.

Heyting, A., Intuitionism, an Introduction, Amsterdam, North Holland Publishing Co., 1956.

Kleene, S. C., "Representation of Events in Nerve Nets and Finite Automata," Automata Studies ed. C. E. Shannon and J. McCarthy, Princeton, Princeton University Press, (1956), 3-21.

Konig, D., Theorie der endlichen und unendlichen Graphen, Leipzig, Akademische Verlagsgesellschaft M. B. H., 1936.

Medvedev, I. T., "On a Class of Events Representable in a Finite Automaton," translated by J. J. Schorr-Kon from a supplement to the Russian translation of Automata Studies (edited by C. Shannon and J. McCarthy), Group report 34-73, 1958, Lincoln Laboratory, Lexington, Massachusetts.

Moore, E. F., "Gedanken Experiments on Sequential Machines," Automata Studies, edited by C. E. Shannon and J. McCarthy, Princeton, Princeton University Press, 1956, 129-153.

Moore, E. F. and Shannon, C. E., "Reliable Circuits using Less Reliable Relays," Journal of the Franklin Institute 262 (1956), 191-208, 281-287.

McKinsey, J. C. C., Introduction to the Theory of Games, New York, McGraw-Hill, 1952.

Mealy, G. H., "A Method for Synthesizing Sequential Circuits, The Bell System Technical Journal 34 (1955), 1045-1079.

Myhill, J., "Fundamental Concepts in the Theory of Systems," WADC Technical Report 57-624, ASTIA Document No. AD 155741, 1957.

Putnam, H., "Decidability and Essential Undecidability," The Journal of Symbolic Logic 22 (1957), 39-54.

Rabin, M. O. and Scott, D., "Finite Automata and their Decision Problems," IBM Journal of Research and Development 3 (1959), 114-125.

Shannon, C. E., "A Mathematical Theory of Communication," Bell System Technical Journal 27 (1948), 379-423, 623-656.

Turing, A. M., "On Computable Numbers, with an Application to the Entscheidungsproblem," Proceedings of the London Mathematical Society Series 2, 42 (1936), 230-265 and 43 (1937), 544-546.

Von Neumann, J., "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components," Automata Studies, edited by Shannon and McCarthy, Princeton, Princeton University Press, 1956, 43-98.

Von Neumann, J., "The General and Logical Theory of Automata," Cerebral Mechanisms in Behavior, New York, John Wiley and Sons, 1951, 1-41.