8. Adaptation of Codings and Representations

To this point the major limitation of genetic plans has been their dependence upon the fixed representation of the structures $\mathfrak{A}$. The object of the present chapter is to show how to relax this limitation by subjecting the representation itself to adaptation. This will be approached by reconsidering representation via detectors $\{\delta_i: \mathfrak{A} \to V_i, i = 1, \ldots, l\}$ (chapter 4) in the light of the comment that detectors can be looked upon as algorithms for assigning attributes (section 3.4). Since algorithms can be presented as strings of instructions, the possibility opens of treating them by genetic plans, much as the strings of attributes are treated. (The mode of action of the genetic operators, of course, puts some unique requirements on the form and interaction of the instructions.) Actually, with a set of instructions of adequate power, we can go much further. We can define structures capable of achieving any effectively describable behavior vis-à-vis the environment. We can do this by setting up algorithms which act conditionally in terms of environmental and internal conditions. In particular, the predictive modeling technique of sections 3.4 and 3.5 can be implemented and subjected to adaptation. The Jacob-Monod "operon-operator" model (see the end of chapter 6) is suggestive in this respect, and we'll look at it more closely after the question of a "language of algorithms" (the instructions and their grammar) has been considered.

1. FIXED REPRESENTATION

Before proceeding to a "language" suited to the modification of representations it is worth looking at just how flexible a *fixed* representation can be. That a fixed representation has limitations is clear from the fact that only a limited number of subsets of $\mathfrak{A}$ can be represented or defined in terms of schemata based on that representation. If $\mathfrak{A}$ is a set of structures uniquely represented by l detectors, each

taking on k values, then of the 2^{k^l} distinct subsets of $\mathcal{A}$ only $(k+1)^l$ can be defined by schemata. However, the question is not so much one of defining all possible subsets, as it is one of defining enough "enriched" subsets, where an "enriched" subset is one which contains an above-average number of high-performance structures.

It is instructive, then, to determine how many schemata (on the given representation) are "enriched" in the foregoing sense. Let $\mathcal{A}$ contain x structures which are of interest at time t (because their performance exceeds the average by some specified amount). If the attributes are randomly distributed over structures, determination of "enrichment" is a straightforward combinatoric exercise. More precisely, let each δ_i be a pseudo-random function and let $V_i = \{0, 1\}, i = 1, \ldots, l$, so that a given structure $A \in \mathcal{A}$ has property i (i.e., $\delta_i(A) = 1$) with probability $\frac{1}{2}$. Under this arrangement peculiarities of the payoff function cannot bias concentration of exceptional structures in relation to schemata.

Now, two exceptional structures can belong to the same schema only if they are assigned the same attributes on the same defining positions. If there are h defining positions this occurs with probability $(\frac{1}{2})^h$. For j exceptional structures, instead of 2, the probability is $(1/2^{j-1})^h$. Since there are $\binom{l}{h}$ ways of choosing h out of l detectors, and $\binom{x}{j}$ ways of choosing j out of x exceptional structures, the expected number of schemata defined on h positions and containing exactly j exceptional structures is

$$(1/2^{j-1})^h\binom{l}{h}\binom{x}{j}.$$

For example, with $l = 40$ and $x = 10^6$ (so that the density of exceptional structures is $x/2^l = 10^6/2^{40} \cong 10^{-7}$), $h = 20$, and $j = 10$, this comes to

$$(1/2^9)^{20}(40!/(20!20!))(10^6!/(99{,}990!10!)) \cong 3.$$

Noting that a schema defined on 20 positions out of 40 has $2^{20} = 10^6$ instances, we see that the 10 exceptional structures occur with density 10^{-5}, an "enrichment" factor of 100. A few additional calculations show that in excess of 20 schemata defined on 20 positions contain 10 or more exceptional structures.

For given h and j, the "enrichment" factor rises steeply as l increases. On the other hand an increase in x (corresponding to an extension of interest to structures with performances not so far above the average) acts most directly on the expected number of schemata containing j structures. With an adequate number of pseudo-random functions as detectors (and a procedure for assuring that every combination of attributes designates a testable structure), the adaptive plan will have adequate grist for its mill. Stated another way, even when there can be no

correlation between attributes and performance, the set of schemata cuts through the space of structures in enough ways to provide a variety of "enriched" subsets. Intrinsic parallelism assures that these subsets will be rapidly explored and exploited.

2. THE "BROADCAST LANGUAGE"

Though the foregoing is encouraging as to the range of partitions offered by a given set of schemata, something more is desirable when long-term adaptation is involved. First of all, when the payoff function is very complex, it is desirable to adapt the representation so that correlations between attributes and performance are generated. Both higher proportions of "enriched" schemata and higher "enrichment" factors result. It is still more important, when the environment provides signals in addition to payoff, that the adaptive plan be able to model the environment by means of appropriate structures (the $\mathfrak{M}$ component of $\mathfrak{A} = \mathfrak{A}_1 \times \mathfrak{M}$ in section 2.1). In this way large (non-payoff) information flows from the environment can be used to improve performance. As suggested in sections 3.4 and 3.5, by a process of generating predictions with the model, observing subsequent outcomes, and then compensating the model for false predictions, adaptation can take place even when payoff is a rare occurrence.

To provide these possibilities, the set of representations and models available to the plan must be defined. Further flexibility results if provision is made, within the same framework, for defining operators useful in modifying representations and models. A natural way to do this is to provide a "language" tailored to the precise specification of the representations and operators—a language which can be employed by the adaptive plan. Some earlier observations suggest additional, desirable properties of this language:

1. It should be convenient to present the representations, models, operators, etc. as strings so that schemata and generalized genetic operators can be defined for these extensions.
2. The functional "units" (cf. detectors, etc.) should have the same interpretation (function) regardless of their positioning within a string, so that advantage can be taken of the associations provided by positional proximity (section 6.3).
3. The number of alternatives at each position in a string should be small so that a richer set of schemata is provided for a given size of $\mathfrak{A}$ (see the comments in the middle of chapter 4).

4. "Completeness," in the sense of being able to define within the language all effective representations and operators, should be provided so that the language itself places no long-term limits on the adaptive plan.

What follows is an outline of one "language" satisfying these conditions. It has the additional property (which will be discussed after the presentation) of offering straightforward representations of several models of natural systems, including operon-operator models, cell-assembly models (section 3.6) and various physical signaling and radiation models.

The basic units of the language are *broadcast units.* Each broadcast unit can be thought of as broadcasting an output signal "to whom it may concern" whenever it detects certain other signals in its environment. For example, a given unit, upon detecting the presence of signals I and I' (perhaps broadcast by other units), would broadcast signal I''. Some broadcast units actually process signals so that the signal broadcast is some modification of the signals detected. In keeping with suggestion (1), broadcast units are specified by strings of symbols. A set of broadcast units, usually combined in a string, will constitute a *device* or structure (an element of $\mathfrak{A}$). Some broadcast units broadcast strings which can be interpreted as (new) broadcast units; broadcast units can also detect the presence of other broadcast units (treating them as signals). Thus, given broadcast units can modify and create others—they serve as operators on $\mathfrak{A}$.

The language's ten symbols $\Lambda = \{0, 1, *, :, \Diamond, \nabla, \blacktriangledown, \Delta, p, '\}$, along with informal descriptions of intended usage, follow. (Exact interpretations for strings of the symbols follow the listing.)

0 These two symbols constitute the basic alphabet for specifying
1 signals. Thus 01011 is a signal which, within the language, has no other meaning than its ability to activate certain broadcast units. Generally a string such as 01011 will be interpreted as the *name* of a particular signal (e.g. the binary encoding of a frequency or amino acid sequence).

* * indicates that the following string of symbols (up to the next occurrence of a *, if any) is to be interpreted as an active broadcast unit.

: This symbol is the basic punctuation mark, used in separating the arguments of a broadcast unit.

For example, *1100:11 designates a broadcast unit which will broadcast the signal 11 one unit of time after the signal 1100 is detected in the unit's environment (see the intended interpretations for strings below).

When this symbol occurs in the argument of a broadcast unit it indicates a "don't care" condition. I.e. any symbol can occur at that particular position of a signal without affecting its acceptance or rejection by the broadcast unit. If the symbol ◇ occurs at the last position of an argument it indicates that any terminal string (suffix) may occur from that point onward without affecting acceptance or rejection.

For example, *1◇00:11 will broadcast 11 if it detects either the signal 1100 or the signal 1000 (or in fact a signal with any of the other symbols at the second position); *100◇:11 will broadcast 11 if it detects any string having the prefix 100, such as 1001 or 10010110 or even 100.

▽ ▽ designates an arbitrary initial or terminal string of symbols (an arbitrary prefix or suffix) when used in the arguments of a given broadcast unit. This symbol gives the unit string-processing capability.

For example, *11▽:▽ will broadcast the suffix of any signal having the symbols 11 as prefix. Thus if 1100 is detected, the signal 00 will be broadcast, whereas if the signal 11010 is detected, the signal 010 will be broadcast. (The resolution of conflicts, where more than one signal satisfies the input condition, is detailed below.) All occurrences of ▽ within a *given* broadcast unit designate the same substring, but occurrences in *different* broadcast units are independent of each other.

▼ A second symbol used in the same manner as ▽. It makes concatenation of inputs possible (see below).

△ This symbol serves much as ▽ and ▼ but designates an arbitrary *single* symbol in the arguments of a given broadcast unit.

For example, *11△0:1△ broadcasts 10 if 1100 is detected, or 11 if 1110 is detected.

p When *p* occurs as the first symbol of a string it designates a string which persists through time (until deleted), even though it is not a broadcast unit.
This symbol is used to quote symbols in the arguments of a broadcast unit.

For example, *11'◇: 10 broadcasts 10 only if the (unique) string 11◇ is detected (i.e., the symbol ◇ occurs literally at the third position); without the quote the unit would broadcast 10 whenever any 3 symbol string with the prefix 11 is detected.

The interpretations of the various strings from Λ^*, the set of strings over Λ, along with the conventions for resolving conflicts, follow.

Let I be an arbitrary string from Λ^*. In I a symbol is said to be quoted if the symbol ′ occurs at its immediate left. I is parsed into broadcast units as follows: The first broadcast unit is designated by the segment from the leftmost unquoted * to (but not including) the next unquoted * on the right (if any). (Any prefix to the left of the leftmost unquoted * is ignored.) The second, third, etc., broadcast units are obtained by repeating this procedure for each successive unquoted * from the left. If I contains no unquoted * s it designates the *null* unit, i.e., it does not broadcast a signal under any condition. Thus

$$p10*11'*\triangle 0{:}1\triangle *{:}11\triangledown{:}11\triangledown$$

designates two broadcast units, namely

$$*11'*\triangle 0{:}1\triangle \quad \text{and} \quad *{:}11\triangledown{:}11\triangledown.$$

There are four types of broadcast unit (other than the null unit). To determine the type of a broadcast unit from its designation, first determine if there are three or more (unquoted) : to the right of the *. If so ignore the third : and *everything to the right of it.* The remaining substring, which has a * at the initial positions and at most two : s elsewhere, designates one of the four types if it has one of the following four organizations.

1. $*I_1{:}I_2$
2. $*{:}I_1{:}I_2$
3. $*I_1{::}I_2$
4. $*I_1{:}I_2{:}I_3$

where I_1, I_2, and I_3 are arbitrary non-null strings from Λ^* except that they contain neither unquoted * s nor unquoted : s. If the substring does not have one of these organizations it designates the null unit. The four basic types have the following functions (subject to the conventions for eliminating ambiguities, which follow).

1. $*I_1{:}I_2$ — If a signal of type I_1 is present at time t, then the signal I_2 is broadcast at time $t + 1$.
2. $*{:}I_1{:}I_2$ — If there is *no* signal of type I_1 present at time t, then the signal I_2 is broadcast at time $t + 1$.
3. $*I_1{::}I_2$ — If a signal of type I_1 is present at time t, then a persistent string of type I_2 (if any exists) is deleted at the *end* of time t.

4. $*I_1{:}I_2{:}I_3$ — If a signal of type I_1 *and* a signal of type I_2 are both present at time t, then the signal I_3 is broadcast *at the same time* t unless I_3 contains unquoted occurrences of the symbols $\{\nabla, \blacktriangledown, \Delta\}$ or singly quoted occurrences of *, in which case I_3 is broadcast at time $t + 1$.

When the final string of any of these units (I_2 for (1), (2), and (3), I_3 for (4)) is interpreted for broadcast, one quote is stripped from each quoted symbol.

The concept of the *state* of a (finite) collection of broadcast units facilitates discussion of potential ambiguities in the actions and interactions of the four types of unit. This state at time t is, quite simply, the set of *all* signals present at time t, including the strings defining devices in the collection, the signals generated by those devices, and the signals generated in the environment of the collection (input signals). Thus the *initial state* is the set of strings used to specify the initial collection of units, together with all signals present initially. If we look again at the definition of type 4 broadcast units, we see that they may actually use signals in the current state to contribute additional signals to the current state (i.e., they can act with negligible delay much as the switching elements of computer theory). For example, given the broadcast units

$$*11\nabla{:}0\blacktriangledown{:}11\nabla\blacktriangledown$$
$$*100{:}100{:}000$$

with environmental (input) signals {100, 110} at $t = 0$ and {100} at $t = 1$, the state $S(0)$ at $t = 0$ is

$$S(0) = \{*11\nabla{:}0\blacktriangledown{:}11\nabla\blacktriangledown,\ *100{:}100{:}000,\ 100,\ 110,\ 000\},$$

and at $t = 1$ it is

$$S(1) = \{*11\nabla{:}0\blacktriangledown{:}11\nabla\blacktriangledown,\ *100{:}100{:}000,\ 100,\ 000,\ 11000\}.$$

The latter signal in $S(1)$, 11000, occurs because the unit $*11\nabla{:}0\blacktriangledown{:}11\nabla\blacktriangledown$ receives both the signal 110 and the signal 000 at $t = 0$, so that $\nabla = 0$ and $\blacktriangledown = 00$, and hence the output 11000 occurs at time $t = 1$. A little thought shows that the instantaneous action of type 4 units does not interfere with the determination of a unique state at each time since type 4 units can add at most a finite number of signals to the current state.

Since the symbols ∇ and $\blacktriangledown$ are meant only to designate initial or terminal strings their placement within arguments of a broadcast unit can be critical to unambiguous interpretation. For types 1 through 4, if I_1 contains exactly one

unquoted occurrence of a symbol from the set $\{\nabla, \blacktriangledown\}$, then that symbol must occur in either the first or last position to be interpreted; otherwise ∇ or $\blacktriangledown$ is treated simply as a null symbol without function or interpretation. If I_1 contains more than one unquoted occurrence of symbols from the set $\{\nabla, \blacktriangledown\}$ then only the leftmost is operative and then only if it occupies the first position. For type 4 the same convention applies to I_2, with the additional stipulation that, if the operative symbol is the same in both I_1 and I_2, then only the occurrence in I_1 is interpreted. Similarly, for types 1 through 4, only the leftmost occurrence of an unquoted $\triangle$ is operative. Moreover, if ∇, $\blacktriangledown$, or $\triangle$ occur unquoted in the output signal of the broadcast unit without interpretable occurrences in the arguments, they are once again treated as null symbols (and are not broadcast). Thus the broadcast unit $*\nabla 11\nabla\triangle 0{:}11\triangle$ "looks for" any signal with a 4 symbol suffix beginning with 11 and ending in 0; for example, the signal 001110 would yield the output 111 one time-step later.

The final source of ambiguity arises when two or more signals satisfy the same argument of a given string-processing broadcast unit. For example, when the state is

$$S(t) = \{*11\nabla{:}\nabla, 111, 1100\}$$

the broadcast unit $*11\nabla{:}\nabla$ could process either 11 or 1100, producing either the output 1 or else the output 00. This difficulty is resolved by having the unit select one of the two signals at random. That is, if there are c signals satisfying a given argument at time t, then each is assigned a probability $1/c$ and one is chosen at random under this distribution. This method of resolving the difficulty extends the power of the language, allowing the representation of random processes.

3. USAGE

The following examples exhibit typical constructions and operations within the "broadcast language":

1. The object is to produce the concatenation of two arbitrary persistent strings uniquely identified by the prefixes I_1 and I_2 respectively. In so doing the prefixes should be dropped and the result should be identified by the new prefix I_3. This is accomplished by the broadcast unit

$$*pI_1\nabla{:}pI_2\blacktriangledown{:}pI_3\nabla\blacktriangledown.$$

2. The object is to generate a sample of the random variable defined by assigning probability $1/n$ to each of the numbers $\{1, 2, \ldots, n\}$. To do this each

string in a set of n persistent strings, say the binary representations of the numbers 1 through n, is prefixed by the same string, say I, which uniquely indicates that the strings are to serve as the data base for the random number generator. When the state $S(t)$ contains these strings and the signal (string) J, the broadcast unit

$$*pI\nabla:J:pI_1\nabla$$

then accomplishes the task, with J signaling that the sample-taking procedure is to be initiated, and I_1 indicating the result. Simple, nonuniform random variables can be approximated by making multiple copies of numbers in the base (so that their proportions approximate the nonuniform distribution). More complex distributions can be handled by using the general computational powers of sets of broadcast units in conjunction with the above procedure.

3. The object is to generate a sample as in (2) but without replacement (the number drawn is deleted from the data base tagged by I). To accomplish this a second broadcast unit is added to the one in (2) giving

$$*pI\nabla:J:pI_1\nabla*pI_1\nabla::pI\nabla.$$

The second unit deletes the string just selected from the data base since just that string is uniquely prefixed with I_1 by the first broadcast unit.

4. A particular substring I_0 is to serve as a special punctuation mark and the object is to cleave an arbitrary string at the first (ith) occurrence of I_0 (if it occurs) in that string. To accomplish this let I be a prefix identifying the string to be cleaved, let I_1 identify the component to the left of I_0 after the cleavage and let I_2 identify the component to the right (including I_0). The following set of broadcast units accomplishes the cleavage at the first occurrence of I_0:

$*J_1:pI_1$ $*J_1:pI\nabla:pI_2\nabla$	Signal J_1 initiates the process and the string which will be developed into the right component is given its initial configuration, i.e., it is "set equal" to the string to be cleaved.
$*pI_2I_0\diamond:J_2$ $*:pI_2I_0\diamond:J_3$	A test is made to see if the punctuation I_0 is a prefix of the current version of the right component. If so signal J_2 is emitted, indicating that cleavage has been accomplished. Otherwise J_3 is emitted, indicating that the test should be repeated one place to the right.
$*J_1:J_4$ $*J_4:J_5$ $*J_5:J_3:J_6$	Signal J_6 indicates that the punctuation test failed exactly two time-steps ago.

$*pI_2\triangle\triangledown:J_6:I_3\triangledown$ $*I_3\triangle:pI_1\triangledown:pI_1\triangledown\triangle$	The left component is readied for a new test by having the leftmost symbol of the "old" right component added to its right end.
$*pI_2\triangle\triangledown:J_6:I_4\triangledown$ $*I_4\triangledown:pI_2\triangledown$	The right component is "updated" by deleting the symbol just added to the left component.
$*pI_2\triangledown:J_6:J_6\triangledown$ $*J_6\triangledown::pI_2\triangledown$ $*pI_1\triangledown:J_6:J_7\triangledown$ $*J_7\triangledown::pI_1\triangledown$	The "old" right and left components are deleted (simultaneously with the "updating" above) so that only the "new" components for testing appear in the state after two time-steps.
$*J_6:J_8$ $*J_8:J_4$	J_4 signals that the punctuation test is to be reinitiated for the "new" components.

To cleave the string at the ith occurrence of I_0, instead of the first, a count must be made of successive occurrences of I_0. Since the signal J_2 signals such an occurrence, this means counting successive occurrences of J_2, restarting the process each time the count is incremented (by issuing the signal J_3) until the count reaches i. The next example indicates how a binary counter can be set up to record the count.

5. The object is to count (modulo 2^n) occurrences of a signal J_2. The basic technique is illustrated by the construction of a one-stage binary counter. The transition function (table) for a one-stage binary counter is

Input at t	*State at t*	*State at t + 1*
J_2 not present	S_0	S_0
J_2 not present	S_1	S_1
J_2	S_0	S_1
J_2	S_1	S_0

The set of broadcast units which realizes this function for an initial state (signal) S_0 is:

$*:J_2:J_0$ $*J_2:J_1$	Makes current condition of input available for calculation of new state (on next time-step).
$*S_0:J_{10}$ $*S_1:J_{11}$	Makes current state available for calculation of new state (on next time-step).
$*J_0:J_{10}:S_0$ $*J_0:J_{11}:S_1$ $*J_1:J_{10}:S_1$ $*J_1:J_{11}:S_0$	Realization of the transition table.

For example, if J_2 occurs at times $t = 1$ and $t = 3$ the sequence of all signals broadcast (the overall state sequence) is:

t	*1*	*2*	*3*	*4*
Input signal	J_2		J_2	
Internal signals		J_1	J_0	J_1
	S_0	S_1	S_1	S_0
		J_{10}	J_{11}	J_{11}

The use of broadcast units to realize the given transition table is perfectly general and allows the realization of arbitrary transition functions (including counts modulo 2^n).

6. Treating the persistent strings as data implies that it should be possible to process them in standard computational ways. As a typical operation consider the addition of two persistent binary integers. The object, then, is to set up broadcast units which will carry out this addition. Let A_1 and A_2 be the suffixes which identify the two strings. The addition can be carried out serially, digit by digit, from right to left. Much as in example (4) the "rightmost" digits are successively extracted by the broadcast units

$$*\nabla\triangle A_1 : I_1\triangle$$
$$*\nabla\triangle A_2 : I_2\triangle$$

These digits, together with the "carry" from the operation on the previous pair of digits, identified by prefix I_3, are submitted as in example (5) to broadcast units realizing the transition table:

I_1 *Addend 1*	I_2 *Addend 2*	I_3 *Carry*	I_4 *Sum*	I_5 *New Carry*
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Successive digits of the sum are assembled by the broadcast unit $*I_4\triangle : pA\nabla : pA\triangle\nabla$ where, at the end of the process, the prefix A designates the sum. A few additional

"housekeeping" units such as $*pA\nabla{:}p\nabla A$, which puts the sum in the same form as the addends, are required to start up the process, keep track of position, etc. The overall process is simply a straightforward extension of techniques already illustrated.

7. As a final example note that any string identified with a suffix I can be *reproduced* by the broadcast unit $*\nabla I{:}\nabla I$. Note additionally that this unit itself has the suffix I! Hence, if we start with this unit alone, there will be 2^t copies of it after t time-steps. By revising the unit a bit, so that its action is conditional on a signal J, $*\nabla I{:}J{:}\nabla I$, this *self-reproduction* can be controlled from outside (say by other broadcast units). By extending this idea, with the help of the techniques outlined previously, we can put together a set of broadcast units which reproduces an arbitrary set of broadcast units (including itself). The result is a self-reproducing entity which can be given any of the powers expressible in the "broadcast language."

At this point it would not be difficult to give the "broadcast language" a precise, axiomatic formulation, developing the foregoing examples into a formal proof of its powers. (For anyone familiar with the material presented, say, in Arbib [1964] or Minsky [1967] this turns out to be little more than a somewhat tedious exercise.) However, our present objectives would be little advanced thereby. It is already reasonably clear that the "broadcast language" exhibits the desiderata outlined at the beginning of section 2. In particular, the broadcast units satisfy the functional integrity requirement (2) in a straightforward way. Consequently, strings of broadcast units can be manipulated by generalized genetic operators with attendant advantages vis-à-vis schemata (see section 6.3 and the close of chapter 7). Moreover a little thought shows that by using the techniques of usage (4) along with those of (2), units can be combined to define a crossover operator which acts only at specified "punctuations" (such as * s or : s or at a particular "indicator" string I). The other generalized genetic operators can be similarly defined. New detectors can be formed naturally from environmental signals (represented as binary strings). For example, a signal can be converted to an argument which will detect similar signals (elements of a superset) simply by inserting "don't cares" ($\Diamond$) at one or more points. Thus, $*E\nabla{:}pI\nabla$ converts any signal with prefix E ("environmental") into a permanent piece of data which can then be manipulated as in usages (4) and (6) to form a new broadcast unit with some modification of the signal as an *argument*.

The collection of broadcast units employed by an adaptive system at any time will, in effect, determine its representation of the environment. Since the units themselves are strings which can be manipulated by generalized genetic operators, strings of units ("devices") can be made subject to reproductive plans and intrinsic

parallelism in processing. More than this, the adaptive plan can modify and coordinate the broadcast units to form models of the environment. By implementing, within the language, the "prediction and correction" techniques for models discussed in Illustration 3.4, we can arrive at a very sophisticated adaptive plan, one which can rapidly overcome inadequacies in its representation of the environment. This approach will be elaborated in the next section. The "broadcast language" already makes it clear that there exist languages suitable both for defining arbitrary representations and for defining the operators which allow these representations to be adapted to environmental requirements.

4. CONCERNING APPLICATIONS AND THE USE OF GENETIC PLANS TO MODIFY REPRESENTATIONS

The broadcast language provides unusually straightforward representations for a variety of natural models. Such representation not only provides a uniform context for comparisons and rigorous study, it also makes clear the "computational" or processing power of the model and its susceptibility to adaptation.

The Britten-Davidson generalization (1969) of the "operon-operator" model serves well to illustrate the point. This is a model for regulation in higher cells; as such it includes many mechanisms not found in the simpler bacterial cells modeled by Jacob and Monod (1961). The model consists of four basic types of gene (see also Figure 14):

1. The *sensor* gene is activated (perhaps via intermediary molecules) by any of various agents (enzymes, hormones, metabolites) involved in inter- and intracellular control. That is, the sensor gene is a detector sensitive to the state of the cell and its environment.
2. The *producer* genes are the specific controls for the actual production of cell structures (membranes, organelles, etc.) and operating agents (enzymes, etc.). They are the output controls of the regulation procedure.
3. Each *integrator* gene is associated with a sensor gene and sends out a specific signal (molecule) to other genes when the sensor is activated. Several integrator genes may be associated with a single sensor, thus allowing the sensor to initiate a variety of signals.
4. The *receptor* gene is a link between integrator genes and producer genes. Each receptor gene is associated with a single producer gene and is sensitive to a single integrator signal. When the signal is received the producer gene is activated. A given producer gene may have several associated receptors.

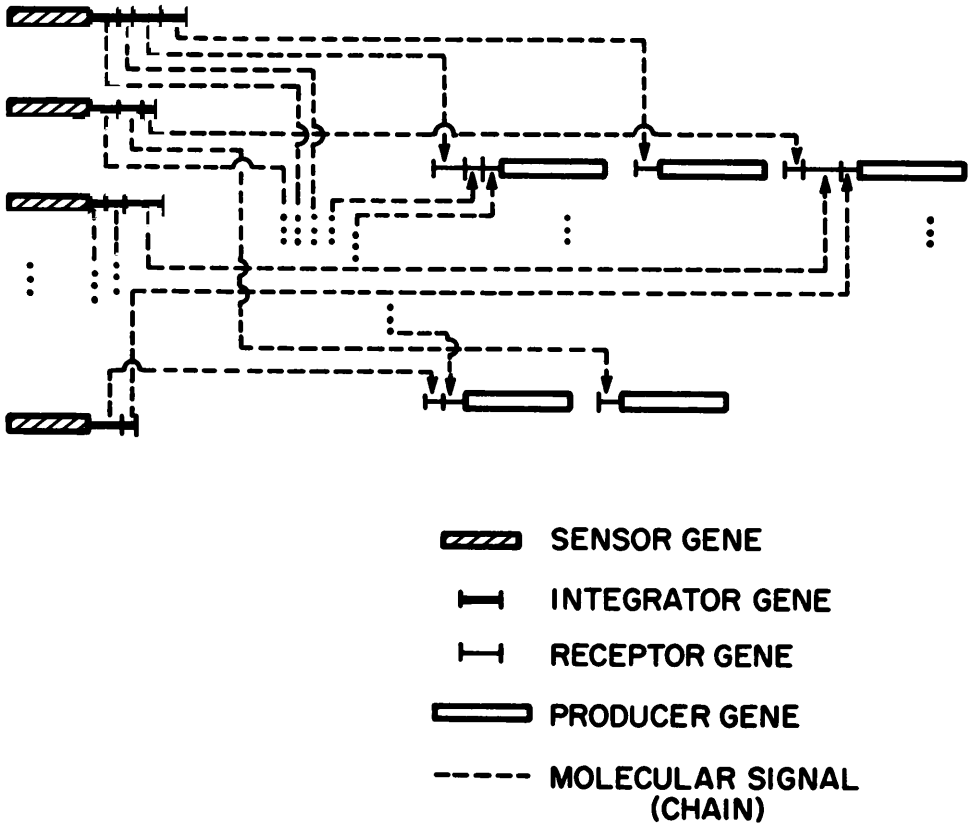

Fig. 14. Schematic of Britten-Davidson generalized "operon-operator" model for gene regulation in higher cells

Translated to the broadcast language a sensor-integrator gene complex $SI_1I_2I_3$ is directly represented by the set of broadcast units $*S{:}I_1{*}S{:}I_2{*}S{:}I_3$. A receptor-producer complex R_1R_2P is similarly represented by $*R_1{:}P{*}R_2{:}P$. If the receptor R_1 responds only to the integrator I_3, say, then $*R_1{:}P$ would be replaced by $*I_3{:}P$. It is apparent, however, that the receptor R_1 *could* be activated by several related signals, say *IA*, *IB*, *IC*, . . . , in which case the producer complex would be represented by $*I\Diamond{:}P$. Similarly a sensor S could be sensitive to any product with an initial radical X so that $SI_1I_2I_3$ could be represented by $*X\Diamond{:}I_1{*}X\Diamond{:}I_2{*}X\Diamond{:}I_3$. Clearly extremely complex feedback loops can be constructed, allowing a great range of conditional actions dependent upon substrate and products already pro-

duced. As an example (simplified for brevity) the sensor-integrator-receptor-producer complex $*X_1\Diamond : I_1A*I_1\Diamond : X_2P*X_2P : I_1C$ maintains production of X_2P if a metabolite with initial radical X_1 ever makes an appearance. In fact, if sensors and receptors can have the same range of sensitivity as the arguments of broadcast units, it is easy to show that there is an appropriate Britten-Davidson model for producing *any* arbitrarily given sequence of products.

A very similar representation can be produced for the lymphocyte immune network, well described by Niels K. Jerne (1973) and presented more technically by M. Sela (1973). In this case the "detectors" are combining sites on antibody molecules produced by lymphocyte cells. The environmental "signals" are invading antigens (e.g., foreign protein molecules). The presence of a detected antigen causes the production of additional lymphocytes (additional "broadcast units," see usage (7) of section 3) which in turn secrete additional antibodies which combine with (and neutralize) the antigens.

A bit further afield the broadcast language can also serve for a straightforward representation of the cell assembly model of the central nervous system (section 3.6). Here the broadcast units are cell assemblies while the "to-whom-it-may-concern" aspect of the broadcast language is reasonably approximated by the large number of neurons (10^3 to 10^4) in other assemblies contacted by *each* neuron in a given cell assembly. (More specific interconnections can be represented by appropriate "tagging" (prefixes) as in section 3.) Then, synaptic "learning" rules which induce fractionation and recruitment in cell assemblies find counterparts in generalized genetic operators which modify representations. Closely associated cell assemblies become the counterparts of tested representational components (cf. schemata), and so on. (The interested reader should consult Plum [1972] for the details of a related model.)

In the context of the broadcast language, the cell assembly model fits smoothly with the predictive modeling technique of section 3.4. A discussion of the latter implementation also gives an indication of how the broadcast language is applied to artificial systems. One implementation which emphasizes the cell-assembly/predictive-modeling fit relies on a set of *behavioral units* which generate action sequences and are modified on the basis of the outcome. Each behavioral unit consists of a population of *behavioral atoms* realized as devices in the broadcast language. If we look back to the search strategies of Figure 6 it is the detectors which have a role comparable to the atoms here. In the broadcast language, the detectors would be broadcast units (or sets of them) with arguments corresponding to the conditions defining the detector. (For example, the atom corresponding to δ_1 in Figure 6 would be activated by any 4-by-4 array with 8 or more dark squares.)

The unit(s) implementing the detector, when activated, would broadcast a signal with an identifying prefix. (For the reader familiar with the early history of pattern recognition these units act much like the demons at the lowest level of Selfridge's Pandemonium [1959].) Other devices would "weight" the signals, "sum" them, and "compare" the result to a threshold (cf. section 7.3) to determine which response signal (from the set of transformations $\{\eta_i\}$) should be broadcast. More generally, the behavioral atoms would be a string of broadcast units with an "initiate" condition C which specifies the set of signals capable of activating the atom, an "end" signal J which indicates the end of the atom's activity, and a "predicted value" signal which is meant to indicate the ultimate value to the behavioral unit of that atom's activation. With this arrangement we can treat the behavioral unit as a *population* of atoms. The atom activated at any given time is determined by a competition between whatever atom is already activated and all atoms having condition C satisfied by a signal in the current state $S(t)$. The higher the predicted value of the atom the more likely it is to win the competition. The object at this level is to have each atom's predicted value V consistent with that of its successor, so that a set of atoms acting in sequence provides a consistent prediction of their value to the behavioral unit. (In this way the atoms satisfy the error correction requirements discussed at the beginning of section 3.4 under element (iii) of a typical search plan.) This object can be accomplished via a genetic plan applied to the population of atoms—the reproduction of any atom is determined by the match between its predicted value and that of whatever atom is next activated. For example, consider two atoms, a_1 with parameters (C, J, V) and a_2 with parameters (C', J', V'), where a_1's end signal acts as a_2's initiate signal. Then $(V' - |V - V'|)$ could be used as a payoff to a_1 for purposes of the genetic plan, since the quantity measures the match between V' and V. The population would then be modified as outlined in section 6.1, new atoms being assigned the predicted value of the successor a_2. That is, the offspring of a_1 would be assigned the predicted value V'. All atoms active since the last actual payoff from the environment, and their offspring, are tagged and their predicted values are adjusted up or down at the time of the next environmental payoff. The adjustment is determined by the difference between predicted value and the actual payoff rate (payoff received from the environment divided by the elapsed time since last payoff). After each environmental payoff the active behavioral unit is subjected to a genetic plan (again as described in section 6.1). The behavioral unit next active (after the environmental payoff) is determined by the winner of a competition among *all* atoms in all behavioral units. The outcome of the competition is determined in the same way as the within-unit competition. Finally, a behavioral unit may be subjected to

competition in the absence of environmental payoff, the probability of such an event increasing as a function of elapsed time since last payoff.

Much more detail would have to be supplied for this implementation of predictive modeling to reach the level of precision earlier given to the description of genetic plans. However the objective here is only to indicate the potential of the broadcast language for predictive modeling with changing representations.

As a final example, note that the world of radiative signals (sound, light, etc.) is susceptible to modeling as a complex broadcast system. In fact one physical realization of devices specified in the broadcast language would assign a unique frequency to each signal and realize broadcast units as a variety of frequency modulation devices.

Even where the broadcast language does not so directly represent extant models, it still supplies a rigorous framework for the description and modification of representations. In particular it makes possible the application of genetic plans to the problem of discovering suitable representations. Because devices are represented by strings and because the functional elements (the broadcast units) are self-defined, the generalized genetic operators of sections 6.2 and 6.3 can be used to modify the devices. Moreover, as indicated in section 3, these operators can themselves be defined within the broadcast language. This makes possible a hierarchy of operators defined with respect to various kinds of punctuation. Thus, one crossover operator could be defined to produce crossing-over anywhere along the string, another could be defined to produce crossing-over only at the symbol : (thereby providing for exchange of arguments between broadcast units), still another only at * (thereby exchanging broadcast units between devices), and so on. The operators so defined introduce a hierarchy of schemata ranging from schemata concerned primarily with varieties of arguments, through schemata concerned with combinations of broadcast units, and on to higher levels of organization (e.g., behavioral atoms, behavioral units, etc.).

Note that for the broadcast language schemata are generally defined with respect to sets of arbitrarily long strings. That is, the set of all devices specifiable in the broadcast language would be the set of *all* strings which can be formed from the ten basic symbols; since a schema designates the set of all strings which match it on its defining positions, each schema designates a countable subset of devices. Using this extension of the notion of a schema, we see that the results of chapters 6 and 7, particularly those pertaining to intrinsic parallelism, extend directly to the adaptation of codings and representations. Since the operators themselves can also be specified within the broadcast language, they can also be made subject to the same adaptive processes.

In sum: This chapter has been concerned with removing the limitations imposed by fixed representations. To this end it is possible to devise languages—the broadcast language is an example—which use strings to rigorously define all effectively specifiable representations, models, operators, etc. Since the objects of the language are presented as strings, they can be made grist for the mill provided by genetic plans. As a consequence the advantages of compact storage of accrued information, operational simplicity, intrinsic parallelism, robustness, etc., discussed in chapters 6 and 7, extend to adaptation of representations.