Holland, J. H. (1980). Adaptive algorithms for discovering and using general patterns in growing knowledge bases. *International Journal of Policy Analysis and Information Systems*, **4**(3), 245-268.

Holland, J. H. (1981). *Genetic Algorithms and Adaptation (*Technical Report No. 34). Ann Arbor: University of Michigan, Department of Computer and Communication Sciences.

Holland, J. H. (1985). Properties of the bucket brigade algorithm. *Proceedings of the First International Conference on Genetic Algorithms and Their Applications* (pp. 1-7). Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Holland, J. H. (1986a). Escaping brittleness: the possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In R. S. Michalski, J. G. Carbonell & T. M. Mitchell (Eds.), *Machine learning, an artificial intelligence approach. Volume II.* Los Altos, California: Morgan Kaufmann.

Holland, J. H. (1986b). A mathematical framework for studying learning in classifier systems. In D. Farmer, A. Lapedes, N. Packard, & B. Wendroff (Eds.), *Evolution, games and learning* (pp. 307-317). Amsterdam: North-Holland. (Reprinted from *Physica 22D*, 307-317.)

Holland, J. H. (1987). Genetic algorithms and classifier systems: foundations and future directions. *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms* (pp. 82-89). Hillsdale, New Jersey: Lawrence Erlbaum Assoc.

Holland, J. H., Holyoak, K. J., Nisbett, R. E., and Thagard, P. R. (1986). *Induction: processes of inference, learning, and discovery*. Cambridge, MA: MIT Press.

Holland, J. H. & Reitman, J. S. (1978). Cognitive systems based on adaptive algorithms. In D. A. Waterman & F. Hayes-Roth (Eds.), *Pattern-directed inference systems.* New York: Academic Press.

Riolo, R. L. (1987a). Bucket brigade performance: I. Long sequences of classifiers. *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms* (pp. 184-195). Hillsdale, New Jersey: Lawrence Erlbaum Assoc.

Riolo, R. L. (1987b). Bucket brigade performance: II. Default hierarchies. *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms* (pp. 196-201). Hillsdale, New Jersey: Lawrence Erlbaum Assoc.

Robertson, G. G. and Riolo, R. L. (1988). A tale of two classifier systems. *Machine Learning*, **3**, 139-159.

Sen, S. (1988). Classifier system learning of multiplexer function. Unpublished report available from Sandip Sen, Department of Electrical Engineering, University of Alabama, Tuscaloosa, AL 35486.

Smith, S. (1980). *A learning system based on genetic algorithms*, Ph.D. Dissertation (Computer Science). University of Pittsburgh.

Stadnyk, I. (1987). Schema recombination in pattern recognition problems. *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms* (pp. 27-35). Hillsdale, New Jersey: Lawrence Erlbaum Assoc.

Waterman, D. A. (1970). Generalization learning techniques for automating the learning of heuristics. *Artificial Intelligence*, **1**, 121-170.

Wilson, S. W. (1985). Knowledge growth in an artificial animal. *Proceedings of the First International Conference on Genetic Algorithms and Their Applications* (pp. 16-23). Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Wilson, S. W. (1987a). Hierarchical credit allocation in a classifier system. *Proceedings of the Tenth International Joint Conference on Artificial Intelligence* (pp. 217-220). Los Altos, CA: Morgan Kaufmann.

Wilson, S. W. (1987b). Classifier systems and the animat problem. *Machine Learning*, **2**, 199-228.

Wilson, S. W. (1988). Bid competition and specificity reconsidered. *Complex systems*, **2**(6), 705-723.

taining default hierarchies once they are formed. Finally, we considered three effector payment rules and found that two of them, sharing and champion, each offer advantages.

We looked at three issues in our discussion of classifier syntax. First, we questioned the necessity of using weighted matching for problems with noisy inputs, and thus doubted the need for that particular extension of the syntax. Second, we suggested that classifier systems are presently unsuited to learn quasi-continuous mappings from input to output, and sketched out a modification that may make this possible. In the modification, the output is computed collectively by the classifiers matching the input, rather than being selected from among their candidate messages. Finally, we noted the awkwardness of fixed-format classifier conditions when the messages to be matched become realistically long and suggested a sparse bit-at-position-naming encoding instead. The consequences could include better linkage of building blocks as well as a new approach to the problems addressed currently by partial matching.

We omitted a number of topics about which something should, eventually, be said, but on which our own thinking is as yet rudimentary: planning and lookahead, the representation of expectations, and ways in which classifier systems can form and carry along histories of their activities and summaries that condense detailed situation information. We also ignored the important topics of classifier speciation and population size.

Our decisions in these matters have been guided by a desire to do good classifier system engineering and create systems that work. Certainly the notion of a classifier system is appealing---how could roving bands of mating, reproducing, fighting, and dying rules fail to captivate our collective imagination? But understanding how such systems can learn effectively requires much effort in rigorous theory formation, mathematical computation, and careful experimentation. Although such activities are sometimes less glamorous than exploring advanced capabilities, more basic questions demand our immediate attention if classifier systems are to move from laboratory curiosity to become working, flexible, adaptive systems. Our parting hope is that this paper has illuminated some small crannies where such labor may pay off.

References

Booker, L. B. (1982). *Intelligent behavior as an adaptation to the task environment*, Ph.D. Dissertation (Computer and Communication Sciences). The University of Michigan.

Booker, L. B. (1985). Improving the performance of genetic algorithms in classifier systems. *Proceedings of the First International Conference on Genetic Algorithms and Their Applications (*pp. 80-92). Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Booker, L. B., Goldberg, D. E., & Holland, J. H. (in press). Classifier systems and genetic algorithms. *Artificial Intelligence*.

Davis, L. and Young, D. (1988). Classifier systems with Hamming weights. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 162-173). San Mateo, CA: Morgan Kaufmann.

Forrest, S. (1985). Implementing semantic network structures using the classifier system. *Proceedings of the First International Conference on Genetic Algorithms and Their Applications (*pp. 188-196). Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Goldberg, D. E. (1983). *Computer-aided pipeline operation using genetic algorithms,* Ph.D. Dissertation. The University of Michigan.

Goldberg, D. E. (1988). *Probability matching, the magnitude of reinforcement, and classifier system bidding* (TCGA Report No. 88002). Tuscaloosa: The University of Alabama, Department of Engineering Mechanics, The Clearing House for Genetic Algorithms.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning.* Reading, MA: Addison-Wesley.

Goldberg, D. E. & Holland, J. H. (Eds.) (1988). [Special issue devoted to genetic algorithms and machine learning]. *Machine Learning***, 3**, 95-245.

Grefenstette, J. J. (Ed.) (1985). *Proceedings of the First International Conference on Genetic Algorithms and Their Applications.* Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Grefenstette, J. J. (Ed.) (1987a). *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms (*pp. 202-209). Hillsdale, New Jersey: Lawrence Erlbaum Assoc.

Grefenstette, J. J. (1987b). Multilevel credit assignment in a genetic learning system. *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms* (pp. 202-209). Hillsdale, New Jersey: Lawrence Erlbaum Assoc.

Grefenstette, J. J. (1988). Credit assignment in rule discovery systems based on genetic algorithms. *Machine Learning,* **3**, 225-245.

Holland, J. H. (1971). Processing and processors for schemata. In E. L. Jacks (Ed.), *Associative information processing (*pp. 127-146). New York: American Elsevier.

Holland, J. H. (1975). *Adaptation in natural and artificial systems.* Ann Arbor: University of Michigan Press.

Holland, J. H. (1976). Adaptation. In R. Rosen & F. M. Snell (Eds.), *Progress in theoretical biology,* 4. New York: Plenum.

One possible way is to switch from *selecting* the output message to *computing* it bit by bit. Suppose we define the basic problem as taking an input string into an output string such that a small change in the input string (one bit, say) results in a small change in the output string. Then let each classifier's output string consist, not of a candidate message, but instead of a weighted string of votes for setting each bit of a *collectively computed* output message. An example classifier might be

1 # 0 # # / 43:1 4:0 15:0 .

The meaning is that if this classifier matched the input, it would vote 43 to set the first output bit to 1, 4 to set the second bit to 0, and 15 to set the third to 0. The system would decide the first output bit by adding up the votes for that bit and choosing 1 or 0 depending on which had the majority. The other output bits would be picked similarly. Payoff---assumed positive---would first be divided by the number of output bits. Then the first of the resulting shares would be split equally and added to the first-bit weights of the classifiers that voted *for* the final decision on the first bit, similarly for the second share, etc. Finally, a payout consisting of a small fraction of each weight would be removed from those weights that had been adjusted. The fitness of a classifier for the genetic algorithm would equal the sum of its weights. When a new classifier was generated, the weights could be inherited in some fashion from the parents, or they could be set nominally, thus saving the need to manipulate them with genetic operators.

A system based on this type of classifier resembles a classical perceptron, with the added advantage that its constituent structures can be modified through the GA. Such a classifier system is likely to have the ''continuous mapping'' property because changing one bit of the input will only change part of the membership of the match set, which will in turn change only a few of the output bits, since only some of the matching classifiers will attend to the changed input bit. Furthermore, the degree of output change per input change should be evolvable, according to the problem, under the GA. We propose investigation of the properties of (continuous) mapping classifier systems both for their potential for extending classifier systems' problem range, and to develop the connections with neural networks.

5.3 Sparse classifier notation

In realistic classifier systems message lengths will be substantial, and many classifiers will contain a high proportion of don't care symbols. The sparsity of such classifiers suggests a classifier syntax that explicitly recognizes information-containing positions and ignores don't cares. There are many ways to implement such a *sparse* notation for classifiers, but in one simple scheme we simply tag condition bits by their name. For example, the rule 1#0## / 00011 could be represented in sparse notation as (1,1) (3,0) / 00011. In this way, unmentioned positions are assumed to be #'s. Explicit don't care characters may also be used and these may be treated as pass-through characters, carrying information from message through the condition to the action.

Going to such flexible codings does require some care, however. Depending on the genetic operators used, there is the possibility of redundancy and conflicts between different parts of the same condition. However, simple rules may be used to determine the outcome of such conflicts during matching. A first-come-first-serve arbitration rule may work well, effectively shielding subsequent positions from expression; this forms a type of intrachromosomal dominance operator. A majority-wins conflict resolution rule may also be useful, although some tie-breaking mechanism is necessary for even splits of 1's and 0's. A probabilistic conflict resolution rule would match a 1 or a 0 with probability equal to the proportion of 1's or 0's in the condition. This latter mechanism provides an interesting counterpoint to Booker's partial matching and other position weighting schemes such as those of Davis and Young (1988) and Stadnyk (1987).

Besides saving space, a sparse rule coding may provide reproductive benefits when acted upon by a genetic algorithm. By eliminating needless don't care characters, building blocks of better rules are likely to be tightly linked. If simple cut and splice operators are adopted together with strength-based reproduction, these building blocks should propagate rapidly, thereby forming better rules more quickly than is possible with a rigid fixed format. Experiments are under way to test this hypothesis in an epistatic optimization domain. If successful, the results should transfer readily to the machine learning arena.

6. Summary

In this paper we have reviewed a number of milestones in the development of classifier systems, indicated a number of important current problems and issues, and suggested some solution directions. At the level of classifier system architecture, we emphasized the twin problems of generation and maintenance of bucket brigade chains. One of our suggestions was to investigate ways to introduce modularity so that all chains could be short. A second suggestion was to reduce intra-chain competition through a merger of the Michigan and Pitt approaches in which the population could contain not only individual classifiers but ''corporations'' of cooperating classifiers.

In our discussion of bidding and payments, we first pointed out that neither the roulette wheel nor noisy auction method of decision-making is ideal under all uncertainty regimes, and suggested an auction with noise dependent adaptively on the variance of received payoffs. Next, we considered the community's travails with the questions of overgeneralization and the formation and maintenance of default hierarchies. On overgeneralization, we pointed to a relatively clear result indicating that the problem can be eliminated if specificity is not factored into a classifier's payout, but also noted that overgeneralization can be greatly attenuated under appropriate taxation and payoff conditions. On default hierarchies we noted regimes in which they can be maintained stably, but acknowledged that their evolution in classifier systems has been rare---except perhaps transiently---and cited an hypothesis that may explain this. We described a more market-like bidding mechanics that may aid in main-

than one classifier. If the nominal payoff amount is *R*, should *R* be shared as above by these classifiers (Holland, 1985), should each classifier receive a full amount *R* (Holland, 1986a), or should *R* perhaps be paid only to the highest scoring classifier in the decision process that led to the selected action---a sort of champion payment scheme (Goldberg, 1989)?

Informal experiments have suggested that the second scheme above is the worst, giving the lowest performance and poorest generalization. Between the other two, payoff splitting and champion, the choice appears to be close. Payoff splitting gives more rapid generalization, but the champion technique may reach higher performance somewhat faster. As noted in Section 4.2, the champion technique's slower generalization rate may help offset forces that tend toward overgeneralization. Perhaps the two techniques can be usefully combined. Further experiments, as well as theoretical investigation, should be done. The question is quite important, as the effectors are the doorway through which payoff enters the system.

5. Classifier syntax

We bring together in this section several questions that involve potential changes in the format or syntax of classifiers. We first discuss whether the existence of noisy inputs in some problems calls for a change in syntax. Next we examine the possibility of using classifier systems to learn continuous mappings. Finally, we look at recodings of classifier conditions for situations in which very long messages can be expected.

5.1 Noisy inputs

There has been relatively little investigation of classifier systems with noisy inputs. Booker (1982, 1985) used inputs in which some bits were fixed but the rest could vary randomly; his system then solved several categorization tasks of increasing difficulty using partial matching. Wilson (1987b) experimented with *payoff noise* in which the environment with some probability reversed its payoff schedule; as long as the noise was moderate, the system evolved the same solution classifiers that it did in the absence of noise. Neither of these investigations dealt with the general situation in which the bits of the input message have some probability of being changed from their original values.

To deal generally with input noise, Davis and Young (1988) suggested that the notion of partial matching should be extended to include weighted matching of individual message bits. Their purpose was to handle noise better than appeared possible with either complete matching or, under noise that varied with bit position, Booker's partial matching. However, their scheme required that the classifier syntax be expanded to accommodate a string of weights, one for each bit. Davis and Young showed that their ''classifier system with Hamming weights'' could achieve optimal performance on a discrimination task having different degrees of noise and noise that varied with taxon position. Unfortunately, they presented neither partial nor complete matching experiments on the same problems.

Experimental comparisons should definitely be made, because the weight string represents considerable computational baggage. Davis and Young argued that a standard classifier system would require very large numbers of classifiers to handle noise as well as the Hamming weight system. In preliminary experiments, however, the first author applied the BOOLE system to Davis and Young's task using the same number of classifiers that they did. The performance for different degrees of level noise was very close to the Hamming system results, especially when BOOLE's decision was based on a strength-weighted ''vote'' of the matching classifiers. In addition, BOOLE evolved classifiers that looked at just one or a few input positions, with #'s for the others---not the elaborate bit combinations Davis and Young felt would be necessary. The question of the ability of standard classifier systems to deal with noise should be thoroughly investigated experimentally and theoretically. At this point, it is far from clear that noise requires a change in standard classifier syntax.

5.2 Continuous mappings

In general, classifier systems produce output messages by *selecting* them based on the satisfaction of classifier input conditions. That is, given a certain input, the system chooses a subset of the set of available classifier messages. Selection permits the system to execute highly discontinuous mappings from input to output: if the input changes slightly---but, from a logical standpoint, critically---the output message(s) can undergo a very large change. Basically the same situation holds under partial matching; in that case the input may have to change somewhat more before the output changes, but when it does, the output change---to new messages---can be equally discontinuous. The ability to implement discontinuous mappings is a valuable characteristic of classifier systems because many tasks require large behavior changes to result from small differences in environmental input.

There are important problems, however, that call for a quasi-continuous mapping from input to output. In such problems, a small input change should produce a small output change---though the mapping itself need not be simple. Many of the sensory-motor mappings in animals have this property. Consider, for example, the mapping from a point in visual space---as encoded by, say, the angles of the two eyes' lines of gaze---to the vector of joint angles of the arm and hand that corresponds to touching that same point with the index finger. Here the overall mapping is highly complex and nonlinear, but it is continuous in that a small change in visual coordinates maps to a small change in joint coordinates. A quite different example is the mapping that simply shifts an input string by one bit. Both of these examples would be awkward to implement with a standard classifier system because a distinct classifier would be required for each possible output (embellishments such as ''pass-through'' would not help). Continuous mappings have largely been the province of artificial neural networks. Can they be achieved with classifier systems?

4.3 Formation and stability of default hierarchies

Holland (1985) argued that default hierarchies should form in classifier systems if classifiers' specificities are factored into their bids: this allows more-specific exception classifiers to outbid, and thus ''protect'', general classifiers when the latter would otherwise make an error. In most systems (again, Wilson's and Grefenstette's excepted) the bid included specificity, yet default hierarchies have clearly not appeared in abundance. As noted earlier, Riolo (1987b) showed, in experiments not using the GA, that default hierarchies could be stable, once formed. Goldberg (1989) reached the same conclusion using a different regime than Riolo's but again not including the GA. The implication is that something about the GA inhibits default hierarchy generation.

Wilson (1988) hypothesized that the bid competition not only protected defaults when they were wrong, it also tended to starve them when they were right, so that defaults, if generated, could not survive. The starvation resulted because the GA could be expected to generate, besides the default, more-specific versions of the default, and these latter classifiers would outbid the default and prevent it from being activated. For example, consider a problem whose solution is ''if the first bit of the input string is a 1, decide 1; if the first bit is a 0, decide 0''. The two classifiers 1## ... /1 and 0## ... /0 solve the problem via a logical partition. On the other hand, the classifiers 1## ... /1 and ### ... /0 solve it via a default hierarchy: if the input is the string ''0 ...'', the general classifier will be correct; if the input is ''1 ...", the generalist will be wrong, but will be outbid by 1## ... /1, which is correct. However, there seems to be nothing to prevent the evolution of 0## ... /0, so that the generalist would be outbid in every situation and thus would not survive. Wilson supported this hypothesis by experiments in which (1) no default hierarchies occurred under the standard algorithm, but (2) a striking one emerged when the algorithm was modified to eliminate the starvation effect.

Wilson's results imply that default hierarchies will be rare when discovery is based on the GA. However, classifier systems do not depend strictly on genetic discovery and may, as we have seen, use other operators, so that the question of the stability of a default hierarchy, once formed, remains important. As noted in Section 4.2, if a classifier pays out a fraction b of its strength S, the strength approaches R/b, where R is the mean payoff to the classifier. Under the usual bidding scheme, bid and payout fractions are equal, so that the bid itself approaches $bS=b(R/b)=R$, independent of b. Consequently the bids of a default and a competing exception classifier tend toward the same value, and, despite its greater specificity, the exception cannot stably beat the default. Previous approaches to this problem have depended, to some extent, on ad hoc fixes to keep the bids different. Here we suggest that stability may ensue under a bidding process that more closely reflects that of auctions in real markets.

In real market economies, individuals examine an item they wish to purchase, decide how much they can afford, and bid up to the point where the buyer's bid meets or exceeds other bids and the seller agrees to sell. In a classifier system, a classifier simply matches a message, yanks out a fixed percentage of its bank balance, and throws the sum at the seller if that amount should happen to be selected in the auction process. To achieve more market-like flexibility, suppose we let a classifier bid up to its standard bid value, $bS = c\sigma S$, but then have the winners *pay out* only the amount *necessary* to beat their competitors. In one simply implemented scheme, each classifier simply broadcasts its standard bid value, a winner is selected, but the winner only pays out an amount equal to that bid by the second best rule. That this or other such *necessity auction* schemes may encourage default hierarchy stability is suggested by the following reasoning.

Let a default and an exception classifier have initially equal strengths $S_d = S_e = S$. Then, in competition, the exception will tend to win since its bid is $c\sigma_e S$ vs. $c\sigma_d S$ for the default. However, under the new payout rule, the winner need only pay out as much as the bid of the second-place bidder---in this case $c\sigma_d S$. Repetition of this contest, with continued victory for the exception, will result in its strength approaching $S_e = R/c\sigma_d$. At the same time, the exception's bid will approach $c\sigma_e S_e = R(\sigma_e/\sigma_d) > R$. What about the default's bid? The default will receive payoff in situations where the exception is not relevant (does not match). Because of its low specificity, the default will almost always be forced to pay out its standard bid value, $c\sigma_d S$, in order to win. Thus the default's strength will tend toward $S_d = R/c\sigma_d$, and its bid will approach $c\sigma_d S_d = R(\sigma_d/\sigma_d) = R$, which, as just seen, is less than that of the exception rule. This margin of bid values should permit the exception rule to win consistently over the default rule when both are active, allowing the stable maintenance of a default hierarchy. Simulations and more analysis are required to back these conjectures.

Default hierarchies are important because they are intuitive, and in many situations reduce the number of rules required by the system. They enlarge the set of acceptable solutions with no change in the size of the set of structures, and they provide broad-brush, transitional rulesets and coverings while the system searches for more correct ones (Holland, Holyoak, Nisbett, & Thagard, 1986; Goldberg, 1989). However, more work is required to determine the conditions under which default hierarchies can both form and persist.

4.4 Effector payment alternatives

In classifier systems, what is paid out by a classifier is shared by the classifiers whose messages the first classifier matched, or, under the implicit bucket brigade, by classifiers in the previously active set. The idea in both cases is to support multiple or parallel activation chains, and to encourage balanced niche formation. The situation is somewhat different when external payoff is received, and there are several options.

When a system takes an action that results in external payoff, that action has been advocated, in general, by more

sion-making mechanisms used in classifier systems and suggest ways they can better deal with varying uncertainty. We next discuss the apparent problem of overgeneralization and then consider a number of questions related to the formation and stability of default hierarchies. Finally, we compare several effector payment schemes. Two definitions will be helpful for clarity. We shall say that a classifier's *bid* is the quantity used by the decision mechanism to determine which classifiers will be activated. Second, we shall use *payout* to mean the total amount, during a cycle, that a classifier pays to the classifiers whose messages it matches and, in the case of detector messages, to the environment.

4.1 Decision techniques

Most classifier systems have made decisions by a stochastic process in which the probability of selection was proportional to a classifier's bid. In the bid competition, the probability was equal to the bid divided by the sum of bids of all classifiers whose conditions were satisfied. In the effector activation competition, the probability of choosing a particular effector state was equal to the sum of bids of classifiers advocating that state, divided by the sum of bids of all classifiers addressing that effector. Goldberg (1983) calls this the ''roulette wheel'' technique. His own choice, however, was a ''noisy auction'' in which each contending bid had added to it gaussian noise having a certain variance, and the classifiers with the largest of the resulting quantities were selected. An advantage of the noisy auction is that by adjusting the noise variance, the degree of determinism in the decision can range from completely deterministic (i.e., based directly on the highest bids) to totally random.

The degree of determinism in the decision is important, and relates to the ''exploration vs. exploitation'' dilemma faced by all inductive systems. If of two classifiers A and B that might be selected, the strength of A is somewhat greater than that of B, then a relatively deterministic decision (for A) should be made only if the system is quite sure that the strengths estimate accurately the resulting payoff. This corresponds to exploitation of current information. On the other hand, if the system is unsure that the estimates are accurate, or if the environment's payoff schedule could be non-stationary, then the decision should be made less deterministically, corresponding to greater exploration of different possibilities. Goldberg (1988) develops the theory of such decision making and shows that neither the roulette wheel nor the noisy auction with fixed variance is optimal over the full range of uncertainty that a system can encounter. He also cites results in the psychological literature that suggest humans adapt the exploitation-exploration balance in their decision procedures to the degree of perceived uncertainty.

To give classifier systems a similar flexibility Goldberg suggests a noisy auction in which the variance of each classifier's bid depends on the variance in its received payoffs. Thus the bids of classifiers whose payoff amount was reliable would be treated quite deterministically, while those of classifiers with uncertain payoff would have a greater stochastic component. The scheme would be implemented by having each classifier keep an estimate of the variance of payoffs to it. We suggest that this *variance-sensitive bidding* is an important topic for investigation. Its effect should be to permit a system to explore alternatives in earlier learning phases much as with roulette wheel decision making, then later, as uncertainty diminishes, to maximize payoff returns through increasingly deterministic decisions.

4.2 Overgeneralization

As noted in the review, classifier systems have a tendency to produce excessive numbers of general and overgeneral classifiers. Some systems, however, have not shown the problem, for example Wilson's Animat and BOOLE, and Grefenstette's state-space traverser. As discussed earlier, the problem can occur when a classifier's specificity is factored into its payout, resulting in higher fixed-point strengths for generalists. In the systems just mentioned, payout was simply a fraction of the strength. Wilson (1988) investigated the question directly, using systems that were identical except that one had specificity in the payout and the other did not. The result was rampant overgeneralization for the former and solid learning for the latter. From this, there would appear to be no good reason for the payout to depend on specificity.

The problem perhaps arose in the first place because of the conceptual elegance of the economic analogy in which a classifier paid its bid to its suppliers. Since the bid contained specificity, then, by the analogy, so should the payout. However, besides biasing the GA in favor of generalists, this policy has other problems. Goldberg (1983) noted that if the payout was a fraction b of the strength, the steady-state strength would approach R/b, where R is the mean payoff to the classifier. If b is constant, then the steady-state payoff is in effect estimated (to within a factor $1/b$) by the strength. If, however, $b=c\sigma$, with c a constant and σ the specificity, then the strength estimates payoff divided by σ. This conflicts with the basic notion that strength, if it is to be a classifier's fitness under the GA, should estimate payoff. The fraction b also determines how heavily recent payoffs are weighted in the current estimate; if b depends on σ, then the weighting is greater for more specific classifiers, to no apparent purpose.

Despite the practical and conceptual difficulties with including specificity in the payout, an argument can be made that at least some of the resulting bias toward general classifiers may be helpful in supporting default hierarchies (Riolo, 1987b). In addition, classifier systems such as those of Goldberg (1983, 1989) and Robertson and Riolo (1988) have included specificity, or a slowly increasing function thereof, in the payout without serious overgeneralization problems. However, Goldberg's programs also used a ''champion'' payoff scheme (see Section 4.4) instead of payoff sharing, and Wilson's (1988) explanation of overgeneralization collapse depends partly on the presence of payoff sharing. In Robertson and Riolo's case, as noted in the review, various means including taxation held generalists in check. The most appropriate conclusion is thus that specificity can be included in the payout and might in some respects be desirable, but a bias toward overgeneralization will occur that must in one way or another be contended with.

this hypothesis, the triggered coupling operator crosses one of those non-coupled classifiers, call it B, with C to produce two new classifiers: one that fires on B's condition and leaves a message M, and one that fires on M and leaves C's message. As a result, there is now a well-defined coupling between B's condition and C's message that represents the hypothesis and can be further tested by the bucket brigade.

Robertson and Riolo's ''*yhwhyhwh*...'' prediction task provides an example. The system should predict ''*w*'' after ''*h*'' if the preceding letter was ''*y*''; otherwise it should predict ''*y*''. If the system can see only one letter at a time, it cannot solve the task unless (1) some classifier on the preceding time-step leaves a message that implies whether ''*w*'' or ''*y*'' was present on that step, and (2) a classifier on the present time-step responds to that message (and the ''*h*'') and makes the right prediction. These classifiers are to be created by triggered coupling out of a classifier (B) that notices the ''*w*'' or ''*y*'' and a classifier (C) that happens by chance to get the right answer and so makes a profit.

Holland (1987) suggests other conditions under which coupling might be triggered in order to induce bucket brigade chains. Although these heuristics deserve investigation, much of their power is wasted under the present bucket brigade architecture. The problem is that events that should be coupled by the operators may, in realistic cases, be many time-steps apart. If, for instance, a system is to leave a room by the same door through which it entered, this coupling will be very hard to establish if the system is also active within the room. As long as the fundamental time-granularity of the bucket brigade is solely that of its most elementary level, significant couplings will be extremely improbable except in elementary tasks.

3.3 Proposal: eliminate long chains

This section has pointed out that under the present architecture, long bucket brigade chains are required for significant action, yet long chains are both hard to support and hard to set up. We propose investigation of methods that will promote modularity in the bucket brigade. Conceptually, all action sequences can be decomposed into modules at various levels of abstraction. For instance, <dress>, <take a taxi>, and <enter theater> might be high level modules of <going to the opera>, whereas <put on shoes> and <tie laces> would be modules---belonging to <dress>---at a lower level, etc. If the bucket brigade were organized so that module sequences at different levels formed separate reinforcement chains, then all such chains would be short, thus making them easier to discover and maintain. There would be further benefits. The essence of a module is that it can be called in many different contexts. Consequently, classifiers forming a module would be more fully tested than classifiers in the present architecture, and populations could potentially be smaller. In addition, execution of a module chain is a unit activity suitable not only for the bucket brigade, but also for Grefenstette's PSP since it forms a well-defined episode (John Grefenstette, personal communication, 1988). Finally, modularity at a range of levels would introduce levels of time-granularity that could permit powerful triggered coupling.

An architecture for a modular or hierarchical classifier system has been suggested by Wilson (1987a), but many schemes are possible. It is probably important to have operating principles that permit a message to persist on the message list over indefinitely many time-steps so as to allow it to name the module that its sub-modules belong to. Further, module chains should probably receive some payoff even if the system as a whole is not rewarded; they should be rewarded for ''doing their job''.

3.4 Proposal: a corporate classifier system

Even with shorter chains and modularity, the population will retain some degree of the cooperator/competitor dilemma noted earlier. Classifiers in a module chain are basically cooperative and stand or fall together. On the other hand, separate modules are in competition if they address similar purposes. Grefenstette (1987b) points out that the Pitt approach directly permits evolution of coadapted---cooperative---sets of rules under the genetic algorithm. It is possible a similar effect could be achieved in a classifier system if, for purposes of reproduction, classifiers could form cooperative clusters.

The idea is that the population would consist not only of single classifiers, but also clusters of classifiers called *corporations*. If two classifiers belonged to the same corporation, they would not compete with each other because the corporation could only be reproduced or deleted as a unit. Corporations would form and break up through a modified crossover operator. The performance and reinforcement components of the classifier system would function almost as usual, i.e., ignoring the clustering. However, for purposes of reproduction, the fitness of a corporation would depend on the strengths of its constituent classifiers in such a way that clustering of cooperators was advantageous over their remaining single. For example, suppose a corporation's fitness were defined simply as the average strength of its member classifiers. This alone might be sufficient to favor clustering of cooperators since, for example, increased fitness stability is generally advantageous, and bucket-brigade transactions within a corporation would leave its fitness unchanged. However, if normal CS effects are not sufficient, pressures can be introduced for the purpose. For example, the reinforcement component could pay a bonus to a classifier that received bucket brigade payoff from a member of the same corporation. Or, transactions not within the same corporation could be taxed. If the system had operating principles that induced both modularity and corporations, the corporations that evolved might well consist of module chains, at least approximately.

4. Bidding and payments

Since the behavior of all classifier systems is driven by reinforcement received from the environment, the mechanisms for distributing that reward internally are primary concerns in classifier system design. The last section was concerned with reward distribution over multiple cycles of the system. In this section, we look at certain questions related to events that take place within a single cycle. We first critique the deci-

basic activity as getting a cup of coffee is extraordinary. At more abstract levels, long chains are equally common: consider the number of steps that can be required to take a trip, make a sale, etc. Under the bucket brigade, however, long chains take a long time to reinforce. Wilson (1987a) and Riolo (1987a) found that the number of repetitions required to fully reinforce the earliest member of an *n* member bucket brigade chain was on the order of 10*n*, or about 100 executions of a chain merely 10 steps long. As it now stands, the bucket brigade model seems practical only for learning problems that can be organized into a small number of steps.

Furthermore, the research has suggested that besides being slow to reinforce, bucket brigade chains are also quite fragile in the sense that earlier members tend to have less strength, regardless of the number of sequence repetitions. This seems partly due to a combination of two effects: (1) the probability of reaching payoff starting with an earlier classifier is less than that starting with a later one, because of stochastic effects at each step; (2) later classifiers tend to have many sequences leading into them, so they are reinforced more often. However, there is probably a further reason. Classifiers in a chain are cooperative: early ones depend on the activation of later ones for their payoff flow, and later ones depend on earlier ones to set up the situations in which they, the later ones, can become activated. At the same time, however, the members of the chain usually compete to make offspring under the genetic algorithm. Of course, all classifiers in the population are in such a competition, but for those making up a chain, the competition is particularly problematic. If, for example, later classifiers tend to be stronger, they will be more successful reproductively, and this will put deletion pressure on the earlier classifiers. But that in turn will feed back and weaken the later classifiers, since the earlier ones set them up. The full ramifications of this instability are not now known, but it is clear that short chains will be less subject to it than longer ones. Grefenstette (1987b) also notes the cooperator/competitor dilemma.

Bridge classifiers are designed to feed environmental payoff almost immediately to the beginning of a chain, independent of its length. Though Riolo showed that these can be made to work once generated, there is no suggestion as to how to bring them into existence without the complication of new special operators. Furthermore, very large numbers of tentative bridges would have to be employed, since any earlier-firing classifier might be the valid origin of a chain.

Grefenstette's profit sharing plan and other epochal algorithms would seem to be an easier way of achieving much the same effect as bridges since they distribute payoff to all earlier classifiers as though by bridge. PSP is not a cure-all, however. As Grefenstette notes, there is a problem in deciding how far back in time to give payoff; it is not always sufficient to stop at the previous reward cycle since the stage may have been set earlier for the present reward. Perhaps more importantly, epochal methods do not depend on a strict causal chain: their vulnerability to freeloaders, parasites, and inappropriately rewarded classifiers has not yet been clearly determined. Finally, the cooperator/competitor dilemma still holds for sequentially firing classifiers under PSP.

3.2 Long chain generation

In the decade since CS-1, there have been few classifier systems in which chains of sequentially activated classifiers have actually been formed. Our review mentioned three: Wilson's Animat, Robertson and Riolo's letter sequence predictor, and the bucket brigade and PSP versions of Grefenstette's (1988) state-space traverser. Of these, only Robertson and Riolo's used internal messages. The three systems learned under conditions of infrequent payoff in which the generation of stage-setting classifiers was required. The discovery of such classifiers remains a difficult problem.

In theory, classifier systems generate whatever rules are needed through the genetic algorithm, but in practice additional mechanisms have often been invoked. There seem to be at least two problems with the usual simple GA used to date. (1) Classifier system tasks typically involve a kind of multimodal optimization in which the system must solve not one, but a number of quite different problems (e.g., finding food under different sets of environmental inputs). It is not clear that classifiers that solve sub-problem A will contain schemata useful for sub-problem B; therefore, unrestricted crossover may result primarily in classifiers that are relevant to no sub-problem. It was to offset this that Booker (and later, Robertson and Riolo) used the GA only among simultaneously active classifiers. (2) In some cases, good solutions to a task may occupy only a very small portion of the space of possible classifiers, so that GA search is difficult. Classifiers close to good ones may not (under exact matching) match, so they are not evaluated and yield no search information. Again, Booker attempted to offset this problem with partial matching. It is interesting that the GA alone was quite successful in two tasks in which every possible classifier matched some problem state and could thus be evaluated. In Wilson's BOOLE system, every possible *L*-bit input string could occur, so any classifier generated received eventual evaluation; the system relied very little on discovery through operators like covering. Similarly, in Grefenstette's state-space system, every possible classifier matched at least one state, and the system evidently did not use covering. In general, however, the states of most tasks form only a small subset of the possible states, and the GA has required auxiliary operators.

The two auxiliary operators used thus far are covering and triggered coupling. Covering creates classifiers that match environmental input and therefore incorporate problem state information that the genetic algorithm may eventually make use of. Triggered coupling is more sophisticated; the logic is as follows. Suppose a classifier C makes a large profit on a certain time step---the incoming receipts under the bucket brigade significantly exceed its outgoing payments. The implication is that C's message was especially appropriate in that situation. But why a profit now, since C simply fired as usual under the bucket brigade? Perhaps there was some unusual condition on the previous time-step that correlated with C's profit. If so, it could not have been detected by any classifier currently coupled into C; their firing was business as usual. Maybe some other classifier on the previous time-step detected the unusual condition. To test

Wilson (1987a) suggested, but did not simulate, a new approach to the long chain problem through his *hierarchical bucket brigade*. This method is designed to induce behavioral hierarchies of classifiers in which modularity keeps all bucket brigade chains short, but overall action sequences can still be long. Separately, Wilson (1987b) used a single-step (no bucket brigade) CS called BOOLE to learn a complex Boolean function. This work also demonstrated parametric control of the system's generalization bias and employed dynamic variation of the crossover rate. Subsequently, Sen (1988) modified Wilson's program and obtained very rapid learning rates, far faster than connectionist networks on the same problem.

Robertson and Riolo (1988) investigated letter sequence prediction and produced the first evidence of the generation, and use by the bucket brigade, of internal messages. The prediction task required internal messages since performance depended partially on memory of letters seen earlier. The best results occurred using a *triggered coupling* operator suggested by Holland (1987). In this method, classifiers active on successive time-steps are sometimes recombined to form a pair of coupled offspring. Addition of this operator improved performance over that obtained when discovery relied only on the genetic algorithm and a *covering* operator much like Wilson's (1985) *create* mechanism. Furthermore, while the GA and covering worked well together, each alone produced substantially inferior results on most tasks. Part of the study investigated the use of large populations of classifiers, finding that performance with 8,000 classifiers was significantly better than that with 500.

Robertson and Riolo provide a useful discussion of several major problems they encountered. One problem was the tendency (often noted informally by others) toward excessive production by the GA of general, and overgeneral, classifiers. This is commonly believed to occur because, in the standard bucket brigade algorithm, the quantity both bid and paid out by a classifier contains the classifier's specificity as a factor. The result is that, other things equal, a classifier's fixed-point strength is inversely proportional to its specificity, so general classifiers are preferentially reproduced. The study limited this problem with taxes that fell more heavily on generalists, and by bias in parent selection. A second problem was that reward-receiving effector classifiers at the end of a chain were much stronger and more numerous than earlier members of the chain, discouraging creation of long chains. The study offset this in several ways: taxes were kept very low; detector messages did not participate in payoff; the number of copies of each type of classifier was limited; and the number of effector messages posted during a time-step was limited. A third problem was that unrestricted mating for crossover led to a large number of lethal or inappropriate classifiers that degraded system performance. The solution was related to Booker's (1982) modification: crosses were allowed only between classifiers that were active at the same time, and therefore tended to have similar structure.

Wilson (1988) investigated several aspects of bidding and payout through experiments with his single-step classifier system. First, he found that overgeneralization could be eliminated by simply removing the specificity factor from the amount a classifier paid out; any power of specificity could remain as a factor in the bid. Second, when this change was made, use of a bid competition and a restricted message list did not improve performance over use of an unlimited message list and no bid competition. Third, the bid competition did not encourage formation of default hierarchies. A reinforcement algorithm somewhat different from the standard was found to support default hierarchy formation. Briefly, the change was to distribute payoff to, and take payout from, all matching classifiers that agreed with the system's decision, leaving the others alone.

Last, Grefenstette (1988) made the first experimental comparison between the bucket brigade and a simplified version of Holland and Reitman's epochal credit allocation scheme. In the latter, termed the *profit sharing plan* (PSP), a constant fraction of the current external reward is paid to each classifier that becomes active since the last receipt of reward. Grefenstette argued that classifier strengths under PSP more accurately predict final rewards than those under the bucket brigade, and his experimental results with a two-dimensional state space showed superior performance for PSP. Building on earlier work (Grefenstette 1987b), he then presented a system called RUDI that combined aspects of classifier systems and the LS-1 approach to rule learning. In RUDI, rulewise apportionment of credit is used to order rules heuristically, thereby promoting tighter linkage of building blocks in subsequent recombination of separate rulesets. This hybrid of the Michigan and Pitt approaches gave results better than either approach alone.

2.4 Summary

Much effort has been devoted to the development of working classifier systems over the past decade, but many open questions remain. In the remainder of this paper, we address some of the shortcomings of current classifier systems and propose alternative approaches in the specific areas of bucket brigade architecture, bidding and payments, and classifier syntax.

3. The bucket brigade architecture

The bucket brigade has the virtue of distributing credit to large numbers of sequentially acting classifiers by means of strictly local transactions among them. Unfortunately, as the review has suggested, the technique has two primary weaknesses: (1) it is difficult to maintain long bucket brigade chains, and (2) they are hard to generate in the first place.

3.1 Long chain maintenance

Maintaining long chains is important, since many systems---animals, certainly---must execute extensive action sequences before receiving environmental payoff. At the level of muscle activations, the number of steps involved in even such a

The first classifier system was Cognitive System One (CS-1) of Holland & Reitman (1978). CS-1 ran a simulated linear maze with external payoff only at the maze ends, so that the correct step-direction had to be learned at each interior point. The system's success was the first example of the generation of rules appropriate to a task under the genetic algorithm, and the effective allocation of credit under conditions of infrequent payoff. Although CS-1 learned under infrequent payoff, it did not do so using the bucket-brigade of today's classifier systems. Instead CS-1 apportioned credit to activated classifiers using an *epochal* algorithm. The epochal algorithm kept track of fairly extensive activation statistics and paid classifiers active since the last payoff event when reward was next received.

Smith (1980) stripped the classifier system of its apportionment of credit system in his study of a system called LS-1 in Waterman's (1970) poker playing task. By so doing, Smith sidestepped many of the knotty questions of credit assignment by requiring genetic evaluation of entire rule sets. This approach contrasts starkly with that of CS-1 where a single rule is taken as the corpuscle of genetic manipulation. Subsequent generations of researchers have joined in the great debate pitting the *Michigan* approach (CS-1-like) versus the *Pitt* approach (LS-1-like), but we will later in this paper make an argument that synthesis of the two approaches is necessary to achieve more flexible learning classifier systems.

Several themes of recurring importance were first addressed by Booker (1982, 1985). First, he assigned non-zero match scores to classifiers whose conditions mismatched messages in some positions, arguing that near-matches provide search information that is ignored under complete matching and also permit the system to respond to situations in which no exact match is obtained. Booker's success with partial matching opened a continuing debate of the merits of various matching techniques. Second, Booker made classifiers that were active in the same situation share the resulting payoff. The consequence was a crowding pressure that dynamically allocated classifiers to distinct sub-problems of the overall task, a technique that has been found effective in several subsequent studies. Last, he restricted classifier mating to pairs that were active in the same situation and thus satisfied a similarity criterion. Booker's idea was to limit the drop in on-line performance due to crosses between unrelated classifiers. The technique was successful, but the issue of mating restriction is not settled since, for example, early in a search it may be desirable to mate promiscuously.

Goldberg (1983) applied classifier systems to two control problems: the centering of a Newtonian object in a one-dimensional space, and generation of a rule set that would cover both normal and exceptional operating conditions on a gas pipeline. There were three main contributions. First, Goldberg's results demonstrated the existence of theoretically predicted *default hierarchies* of classifiers (Holland, 1981, 1985) in which, in the simplest case, a default general rule covers the normal situations encountered but is outbid by one or more exception rules that take control in situations where the default would be incorrect. Default hierarchies have been a continuing theme since. Second, Goldberg introduced the *noisy auction*, an alternative to the *roulette wheel* method for calculating bids that offers greater control possibilities and has led to further scrutiny of bidding techniques (Goldberg, 1988). Finally, his work was the first application of a CS to a physical---albeit simulated---system.

Although Goldberg showed the existence of default hierarchies, his work did not consider infrequent reward. Wilson's (1985) Animat system was the first to demonstrate the bucket brigade---though a simplified one---under infrequent payoff conditions. The simplification was to omit the posting and matching of messages; classifiers controlling the system on the current step simply passed a fraction of their strength to the classifiers active on the previous step. Using this *implicit bucket brigade*, the system evolved classifier chains up to four or five steps long. Wilson introduced two other procedures that aided rapid learning in this food-finding task. First, when the system could not match an environmental input, a matching classifier was simply created using an action chosen randomly or by a form of lookahead. Second, each classifier stored an estimate of the average number of steps between its activation and the finding of food; the estimate was combined with strength to form the classifier's bid. This encouraged the formation of paths that were both remunerative and short.

Forrest (1985) investigated the use of a classifier system to implement a subset of the KL-ONE semantic net language. Although the study did not consider learning and would thus seem misplaced in this paper, her work represents an important milestone in the development of classifier systems. Classifier systems have sometimes been criticized as being too primitive, unable to emulate the cognitive models and relationships of symbolic AI systems. Forrest debunks such symbol chauvinism in a clearly stated existence proof. Whether such modeling can be learned remains an open question, and learning such complex structures is a primary thrust of the present paper.

2.3 Since 1985

Riolo (1987a,b) addressed two issues that appeared problematic for the bucket brigade: long chains and default hierarchy instability. Early classifiers in a long chain are difficult to reinforce because of the number of trials required to transport strength up the chain and because of losses along the way. Using simulated chains, Riolo showed that so-called *bridging classifiers* (Holland, 1985) can carry strength from one end of the chain to the other in a single time step.

Factoring a classifier's specificity into its bid had been observed to lead to instability in the relative bids of default and exception classifiers in a default hierarchy. Riolo showed this could be corrected by further biasing the bid competition in favor of specialists. Both studies showed that viable structures can be maintained once appropriate classifiers are generated. However, the studies did not include the genetic algorithm or other discovery mechanism, so the question of *formation* of bridges or default hierarchies was left open.

From Proceedings of the Third International Conference on Genetic Algorithms (1989), *Morgan Kaufmann.*

A Critical Review of Classifier Systems

Stewart W. Wilson
The Rowland Institute for Science
Cambridge, MA 02142

David E. Goldberg
The University of Alabama
Tuscaloosa, AL 35487

Abstract

The current state of classifier system development is examined with emphasis on challenges and unsolved problems. Suggestions related to the bucket-brigade architecture, the mechanics of bidding and payments, and classifier syntax follow a review of past research.

1. Introduction

Depending on the paper you choose to mark the beginning of their development, classifier systems (CSs) have just passed either their eighteenth or eleventh birthday. Age 18 is obtained if you start the clock with ''Processing and Processors for Schemata'' (Holland, 1971); age 11 is calculated if you choose ''Cognitive Systems Based on Adaptive Algorithms'' (Holland & Reitman, 1978) as your milestone of choice. Some may find it surprising that this classifier system ''new'' kid on the block has actually been hanging around for a decade or two. Over the last decade especially, classifier systems have received increasing attention from researchers interested in developing flexible machine learning systems. Yet, as with many children approaching adolescence, these research efforts have reached a developmental crisis---a crisis that must find at least partial resolution before further strides are likely to be successful.

In this paper, we examine the current state of classifier system development with emphasis on areas that we believe contain the most important unsolved problems and, therefore, challenges. Adopting the perspective that classifier systems are intended as a general approach to the induction of needs-serving behavior in uncertain environments, we first discuss past research that has aided in moving the field toward this goal. We then focus on three important areas where our understanding is incomplete: the architecture of the bucket-brigade, the mechanics of bidding and payments, and the syntax of classifiers. While reviewing these topics, we offer suggestions that may solve some of the identified problems or stimulate others in their own search for solutions.

We assume that readers of this article are familiar with basic CS structures and algorithms. If this is not the case, several publications now exist that describe classifier systems in considerable detail. Holland's (1986a) description is sometimes regarded as the standard reference. Goldberg (1989) provides a teaching introduction to classifier systems and reviews or enumerates much of the research published prior to the book's publication. Further material may be found in Grefenstette (1985, 1987a), Booker, Goldberg, and Holland (in press), Goldberg and Holland (1988), and a growing number of other sources.

2. Milestones

Our present knowledge of classifier systems is based primarily on theoretical and experimental studies over the last decade. The experimental work tested variations, adjustments, and extensions of the basic theoretical framework developed by Holland (1975, 1976, 1980, 1981, 1985, 1986a, 1986b, 1987). To save space, we have been selective in choosing studies for the following review. They are arranged chronologically, but we hope at the same time to bring out major themes that several studies have addressed.

2.1 CS prehistory

Holland foreshadowed the coming of classifier systems in his 1971 paper ''Processing and Processors for Schemata''. In a sequence of four prototype proposals he progressed from a simple stimulus-response schema processor to a sophisticated automaton with anticipatory modeling. These suggestions led to the call (Holland, 1975) for the creation of the *broadcast language*, a Post-like production system combining computational completeness, genetic operator amenability, and sophisticated pattern matching and processing capability. Neither broadcast language nor schemata processors made it to the stage of experimental testing, but both helped shape the coming of the first classifier system, CS-1.

2.2 1978 to 1985