Two Views of Classifier Systems

Tim Kovacs

Department of Computer Science
University of Bristol
Bristol BS8 1UB, England
kovacs@cs.bris.ac.uk
http://www.cs.bris.ac.uk/~kovacs

Abstract. This work suggests two ways of looking at Michigan classifier systems; as Genetic Algorithm-based systems, and as Reinforcement Learning-based systems, and argues that the former is more suitable for traditional strength-based systems while the latter is more suitable for accuracy-based XCS. The dissociation of the Genetic Algorithm from policy determination in XCS is noted, and the two types of Michigan classifier system are contrasted with Pittsburgh systems.

1 What is a Learning Classifier System?

What a Learning Classifier System (LCS) is seems contentious, to the extent that discussion of this issue dominated the First International Workshop on Learning Classifier Systems (IWLCS-92). In his report on the workshop Robert Smith paraphrased Lashon Booker as follows:

> "The LCS is usually described as a *method*: a set of algorithmic details that define a way to solve a class of problems. However, in many ways the LCS is more of an *approach*: a set of conceptual details that define a certain direction for developing methods. Therefore, the defining issues for the LCS are not necessarily algorithmic, but conceptual. The central problem addressed by the workshop's discussions was to clarify these defining, conceptual issues." [24] p. 2.

That conceptual issues remained a concern for classifier systems in 2000 is indicated by the inclusion of a series of 11 short essays under the title "What is a Learning Classifier System?" in a recent publication [15].

One conceptual issue addressed here is whether classifier systems are best characterised as evolutionary systems or Q-learning-like systems. Of course classifier systems traditionally incorporate both evolutionary and Reinforcement Learning (RL) algorithms. Given their hybrid nature it seems inappropriate to attempt to cast them strictly as one or the other, and this is not the aim of this work. There are, however, conflicting views of the relationship and relative importance of the classifier system's components, as the following quotes should suggest.

In Holland and Reitman's discussion of the two learning subsystems in the first classifier system, CS-1 [16], it is clear that they consider the Genetic Algorithm (GA) the primary learning system and the credit assignment system decidedly secondary:

> "The second process [credit assignment] is a form of simple learning; after a series of actions, it stores in memory information about the consequences of these actions. The third process [the GA] is a more complex learning process . . . the novelty of the model [the LCS] is not so much in the performance or simple learning processes [credit assignment], but rather in the process that changes memory [the GA]." [16] p. 470.

Although the quote above is from 1978, this view is not confined to early work on classifier system, as the following quote from 1998 suggests:

> "The learning classifier system (LCS) is an application of the genetic algorithm (GA) to machine learning." [9] p. 299.

However, the classifier systems literature also contains contrasting views, in which the role of the GA is not emphasised so much. For instance:

> "In many ways, the LCS can be thought of as a GA-based technique for grouping state-action pairs." [25] B1.5:8.

Although classifier systems are still "GA-based" in the above, the GA is clearly not the whole story. Grouping state-action pairs is not the ultimate goal of a classifier system; something must be done with these groups. Finally, a quote which appears to minimise the role of the GA, at least in XCS:

> "In XCS, the 'discovery component' does not actually discover new classifiers, if one means classifiers that 'do the right thing' in the situations they match, i.e., correctly connect conditions with actions.
> Instead, the discovery component searches along the specificity-generality axis for classifiers that are maximally general while still accurate." [44].

These quotes are meant to suggest a diversity of opinion in the literature, and that the issue of how to characterise classifier systems is worthy of further examination. In the following section the difference between classifier systems and genetic algorithms is briefly examined, following which different types of classifier systems and different views of them are considered.

2 Two Views of Classifier Systems

Although I was unaware of the diversity of opinion regarding the nature of classifier systems when I first began working with them in 1996, I soon became concerned that I was unsure of the difference between a genetic algorithm and a classifier system. I had seen the LCS described as a combination of a production system, rule discovery system and credit assignment system. I reasoned that since the rule discovery system typically *is* a genetic algorithm, the LCS must be something more, since it has two additional components. However, I decided the credit assignment system was just what we call the fitness function of a genetic algorithm. Granted, credit assignment in an LCS was more complex than the examples of function optimisation with a GA which I had seen, but it was still a kind of fitness function. This left the production system as the real difference between the LCS and GA. But since the production system is conceptually straightforward – its task is simply to apply the rules when appropriate – a classifier system seemed to be just a way of applying a GA to certain kinds of problems. Certainly we need to wrap the GA up with a little machinery (the production system and a special kind of fitness function) to interface it with the problem, and perhaps the GA needs a little help in the form of operators like covering (see, e.g., [13,4,43]), but the LCS seemed to be essentially a GA.[1]

This is a view which I still think is consistent with Holland's intentions, and those of many others. Classifier systems have, after all, been described as *Genetics-Based* Machine Learning (GBML) systems [11].

Less-genetic Classifier Systems The view of a classifier system as essentially a GA is somewhat extreme, and other less extreme views exist. In addition to the GA, a classifier system may contain rule discovery mechanisms such as covering, triggered chaining [22], bridging [14,21], and corporate linkage [45,39]. In such systems the GA is just one component of the rule discovery system, although perhaps an important one. However, some LCS emphasise the use of non-genetic operators more heavily than others, and in some cases the GA is even considered a 'background' operator [3].

Non-genetic Classifier Systems That a classifier system is essentially a GA is flatly contradicted by the considerable recent work on LCS which use alternative rule discovery systems. In hindsight, there seems no justification for insisting on the use of GAs as opposed to other evolutionary algorithms. Alternatives were suggested some time ago [42,40,41,25] and some recent work has indeed used Genetic Programming rather than Genetic Algorithms [19,1]. What's more, a significant amount of recent work has been on systems which contain no evolutionary algorithms [30,32,31,33,34,5,6,35–37,7]. If we accept such systems as classifier systems (as is the norm, e.g., work on such systems has appeared at the IWLCS workshops), we are dealing with a much broader concept than that of a GA and some wrapping. Unfortunately, discussion of this trend lies outside the scope of this work.

Linking Classifier Systems and Mainstream RL A second trend which breaks from the view of classifier systems as GA-based systems is that which

[1] A more detailed account of the differences between the two would be desirable, but must be deferred to another work.

seeks to link classifier systems and mainstream reinforcement learning. RL has made great strides since the introduction of classifier systems, and it is clear that most classifier systems are RL systems, that they address many of the same issues addressed by other RL systems, and that there is much to be gained from integrating classifier systems with mainstream RL. The need to bridge classifier systems and mainstream RL appeared to be the consensus during the discussion at IWLCS-99.

The GA-view and RL-view This leaves us with two contradictory views of what a classifier system really is, what we might call the *GA-view* – that the LCS is essentially the application of a GA to a problem – and the *RL-view*; that the LCS is a kind of RL system, i.e., a Q-learning-like system in which the GA is (or may be) a component, but in which many of the interesting issues are to do with credit assignment. The two views place different emphasis on different subsystems: according to the GA-view, the GA, and issues relating to it, are of primary importance, while the RL-view places greater importance on credit assignment. The existence of two alternative views begs an important question: does a classifier system solve problems using evolutionary means, or does it solve them in the way non-evolutionary RL systems do?

One aim of this work is to recognise and publicise the existence of these alternative views, since they seem under-recognised, particularly in the literature. Another aim is to clarify these views, and to justify the RL-view of (some) LCS. Significantly, the RL-view focuses on XCS, which I have argued differs fundamentally from Holland's LCS, to the extent that it more closely resembles mainstream RL systems such as tabular or neural network-based Q-learners [18].[2]

Without a good, basic understanding of RL, the distinction between the GA and RL views of LCS is likely to be unclear. Unfortunately a review of RL algorithms is impossible here, but readers interested in the subject are encouraged to consult [38], which should be required reading for anyone wishing to apply LCS to RL problems. I believe the future of LCS research (for sequential tasks) is heavily grounded in the RL-view; the most important issues to be addressed in LCS research are those which are and will be addressed in RL. This is not to marginalise evolutionary approaches to RL, but to say that they too will benefit from understanding of non-evolutionary approaches.

3 Two Classifier Systems

Two major types of classifier systems (and hybrids of them) appear in the literature. In *Pittsburgh*-style (Pitt) classifier systems (e.g., LS-1 [27–29]) the GA operates on chromosomes which are complete solutions (entire sets of rules), whereas

[2] Despite this, XCS originated and has been studied exclusively within the LCS community, and is by far most strongly integrated with the LCS literature. Much better integration with mainstream RL awaits.

in the more common *Michigan*-style LCS chromosomes are partial solutions (individual rules) (see [11] for further explanation). Some hybrid Pittsburgh-Michigan systems exist (e.g., [12,10,17]).

Although this difference may seem minor, Michigan and Pittsburgh systems are really quite different approaches to learning, as we will see in section 4. The differences between the two may account for some, though not all, of the difficulty in characterising classifier systems. In this work we will consider Michigan systems, although their relation to Pittsburgh system will be highlighted.

To make matters more concrete we will consider two particular Michigan classifier systems: XCS and SBXCS (Strength-Based XCS). XCS, introduced by Wilson in 1995 [43], is the most popular classifier system to date, and is well-documented elsewhere (e.g., [43,8]). Its primary distinguishing feature is its *accuracy-based fitness*; in the GA, it bases the fitness of a rule on the accuracy with which it predicts environmental reward. This contrasts with traditional *strength-based fitness*, in which the fitness of a rule depends on the magnitude of the reward it receives. This difference has a surprising number of consequences [18].

SBXCS [18] is XCS's strength-based twin; that is, it is a version of XCS to which the minimal changes have been made in order to make it a strength-based system. SBXCS is a functional classifier system, quite capable of tasks such as the 6 multiplexer [18], and generally similar to ZCS [42]. SBXCS's value lies in the fact that while it is algorithmically very similar to XCS, it has rather different capacities. This combination of traits allows us to attribute the difference in capacity to the responsible mechanisms more easily.

Estimating Strength/Prediction A classifier system contains two major processes which require estimates of the value of rules: action selection and rule discovery. Both XCS and SBXCS base their value estimates on the rewards they receive, though they differ in how they do so. To begin, both calculate the *strength* (called *prediction* in XCS) of a rule using the Q-learning update, adapted for classifier systems [43]. In non-sequential tasks the update is:

$$p_j \leftarrow p_j + \beta(P - p_j) \tag{1}$$

where p_j is the prediction of rule j, β is the learning rate and P is the payoff (reward) from the environment [43]. The update is applied to the set of rules which match the current input and advocate the action taken by the system, called the *action set* [A]. XCS and SBXCS differ in what follows the prediction update, as illustrated in figure 1, and explained in the following sections. Full specifications of XCS and SBXCS appear in [18].

3.1 SBXCS

In order to estimate the value of taking a given action in response to the current input, SBXCS consults all the rules which match the current input and advocate that action. $S(a_i)$, the strength of action a_i, is defined as:

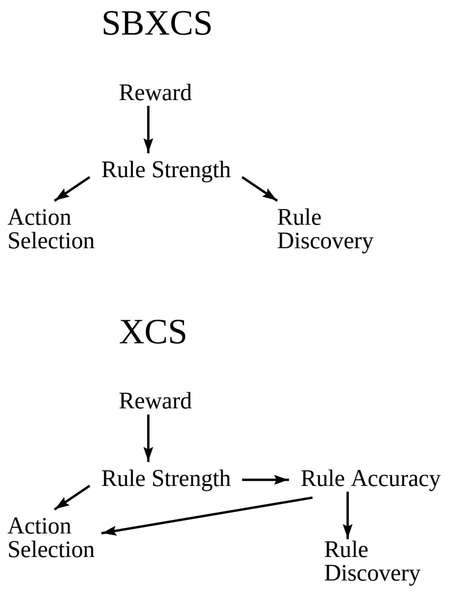

Fig. 1. How XCS and SBXCS use rewards to weight rules in action selection and rule discovery. The arrows indicate the flow of information.

$$S(a_i) = \sum_{c \in [\mathrm{M}]_{a_i}} p_c \cdot numerosity(c) \tag{2}$$

where $[\mathrm{M}]_{a_i}$ is the set of matching rules advocating action a_i, and $numerosity(c)$ is the number of copies of rule c in the population [43]. In short, the strength of action a_i is the total of the strengths of all the rules which advocate it. Significantly, if $[\mathrm{M}]_{a_i}$ is empty action a_i cannot be selected.

Once SBXCS has calculated the strength of each action $S(a_i)$, any of a great number of methods can be used to select an action, e.g., random selection (pure exploration mode [43]), deterministic greedy selection (pure exploitation mode [43]), or ϵ-greedy selection [38], to name but a few.

3.2 XCS

In XCS, calculation of the advocacy for an action is more complicated than in SBXCS, in that we must initially calculate rule fitnesses using the following series of updates. First we update the prediction error ε_j of each rule j in the action set:

$$\varepsilon_j \leftarrow \varepsilon_j + \beta\big(|P - p_j| - \varepsilon_j\big) \tag{3}$$

Next we calculate the *accuracy* κ_j of each rule:

$$\kappa_j = \begin{cases} 1 & \text{if } \varepsilon_j < \varepsilon_o \\ \alpha(\varepsilon_j/\varepsilon_o)^{-v} & \text{otherwise} \end{cases} \tag{4}$$

where $0 < \varepsilon_o$ is the *accuracy criterion*, a constant controlling the tolerance for prediction error. Rules with $\varepsilon < \varepsilon_o$ are considered to be equally (and fully) accurate. The *accuracy falloff rate* $0 < \alpha < 1$ and *accuracy exponent* $0 < v$ are constants controlling the rate of decline in accuracy when ε_o is exceeded.

Once the accuracy of each rule in [A] has been calculated, we calculate the relative accuracy κ' of each rule:

$$\kappa'_j = \frac{\kappa_j \cdot numerosity(j)}{\sum_{x \in [A]} \kappa_x \cdot numerosity(x)} \tag{5}$$

Finally, we update each rule's fitness F_j towards its relative accuracy:

$$F_j \leftarrow F_j + \beta(\kappa'_j - F_j) \tag{6}$$

In summary, the XCS updates treat the strength of a rule as a prediction of the reward to be received, and maintain an estimate of the error ε_j in each rule's prediction. An accuracy score κ_j is calculated based on the error as follows. If error is below the accuracy criterion threshold ε_o the rule is fully accurate (has an accuracy of 1), otherwise its accuracy drops off quickly. The accuracy values in the action set [A] are then converted to relative accuracies (the κ'_j update), and finally each rule's fitness F_j is updated towards its relative accuracy. To simplify, in XCS fitness is inversely related to the error in reward prediction, with errors below ε_o being ignored entirely.

When it comes to selecting actions, XCS weights the predictions of the matching rules by their fitnesses in order to obtain the system prediction $P(a_i)$ for each action a_i:

$$P(a_i) = \frac{\sum_{c \in [\mathrm{M}]_{a_i}} F_c \cdot p_c}{\sum_{c \in [\mathrm{M}]_{a_i}} F_c} \tag{7}$$

Then, as in SBXCS, any action selection method may be used on the $P(a_i)$ values.

3.3 Representation in XCS and SBXCS

XCS tends to find rules which form *complete maps* of the input/action/reward space because 1) the fitness of a rule does not depend on the magnitude of the reward it receives, so no region is neglected simply because it does not generate high rewards, and 2) pressure towards diversity in the rule population drives it towards covering the entire space [43, 18].

In contrast, SBXCS (and other strength-based LCS) tend towards *best action maps*, in which only the most highly rewarded action for each state is represented. This is a form of *partial map*, since not all actions are advocated. In practice, SBXCS does not actually represent only the best action for each state, but it does tend to represent only a subset of the available actions. This occurs simply because the fitness of a rule *is* proportional to the magnitude of the reward it receives [18].

The difference between best action and complete maps is illustrated in figure 2 which shows the 3-bit multiplexer function represented using two sets of rules; in the centre a partial map, and on the right a complete map.

A	**B**	**C**	**Output**
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

0 0 # → 0
0 1 # → 1
1 # 0 → 0
1 # 1 → 1

0 0 # → 0
0 0 # → 1
0 1 # → 0
0 1 # → 1
1 # 0 → 0
1 # 0 → 1
1 # 1 → 0
1 # 1 → 1

Fig. 2. Truth table (left), best action map (centre) and complete map (right) for the 3 multiplexer.

4 The Policy's the Thing

Section 1 outlined two views of classifier systems: the GA-view – that the LCS is essentially the application of a GA to a problem – and the RL-view, that the LCS is some kind of RL system, that is, something like Q-learning, perhaps with some kind of evolutionary component. Which view is correct? Or, more likely, which view is most appropriate? Perhaps it depends on the classifier system in question. Are the two views compatible? The following sections suggest some answers.

Classifier systems for reinforcement learning have the same goal as any RL system: to maximise return. To do so, an RL agent must find an optimal policy.[3] However, an important feature of classifier systems – many would say the whole point of using them, rather than, say, tabular Q-learning – is their in-built capacity to exploit environmental regularities. We could say the primary tasks an LCS faces are finding good policies and finding useful generalisations while doing

[3] Return is a generalisation of the notion of environmental reward. A policy is a mapping from each state to an action; an optimal policy is one which maximises return. See [38].

so. Consequently, we can address the GA- and RL-view question by considering how an LCS finds (selects, determines) policies, and how it finds generalisations.

In fact, we will overlook the issue of finding generalisations. They are expressed by #s in rule conditions, and rule conditions are the concern of the rule discovery system. In both SBXCS and XCS the main component of the rule discovery system is the GA; in other words, in both systems it is up to the GA to find useful generalisations. Of course, for the GA to do so the credit assignment system must provide it with useful information; it must give useful generalisations higher fitness than less useful or less general conditions. But let's ignore generalisation and focus on the learning of policies. Two broad approaches have been mentioned: evolution and Q-learning-like RL. Which approach does a (Michigan) classifier system use? Let's consider SBXCS and XCS in turn.

4.1 How SBXCS Determines Policies

We know that, to a first approximation, SBXCS's rule population determines its policy, and that SBXCS's GA evolves its rules, so we could say that SBXCS's GA determines its policy. But this is a superficial analysis, and in fact we could apply the same reasoning to XCS. Let's consider SBXCS's operation in more detail.

A policy is a mapping of states to actions. A rule population may not specify a unique policy – often conflicting rules occur, and some states may have no matching rules. Given this, how can we tell what a classifier system's policy is?

Deriving a Policy from a Value Function: Tabular Q-Learning Let's consider a tabular Q-learning system. It maintains a value function, and just as a set of rules is not a policy, neither is a value function.[4] (A set of rules can represent a value function, but it can also represent conflicting actions for the same state; it is more general than a policy.) From its value function, a tabular Q-learner derives a policy *as needed.* That is, each time the system must select an action it consults its value function for the current state, and applies some action selection method (e.g., ϵ-greedy selection). Since the value function is constantly being updated, so is the policy. The point here is that there is no explicitly represented policy; it is generated as needed.

Deriving a Policy from a Value Function: SBXCS A classifier system selects actions based on the currently matching rules (see equations (7) and (2)), using some action selection method, in the same way a tabular Q-learning system does based on its value function. From this description an LCS operates just like a tabular Q-learner. But in some cases there's a difference. Recall from section 3.3 that SBXCS maintains partial map representations, and that consequently sometimes (in fact, usually) some of the actions available in a state have no

[4] A value function is a mapping from each state (or state-action pair) to an estimate of its value. See [38].

value estimate. SBXCS (following XCS) cannot select actions whose values are not estimated (i.e., which are not advocated by a matching rule).

In contrast, in a tabular Q-learning system, all actions are represented and can be selected. Is this significant? Yes, because the presence or absence of rules is determined by the GA. By determining which actions are represented, the GA influences action selection.

4.2 How XCS Determines Policies

XCS and SBXCS can be configured to select actions using the same method (e.g., ϵ-greedy selection), and neither can select actions which no matching rule advocates. However, XCS's strong tendency towards complete maps means that, typically, all actions *are* represented by some rule. The significance is that in XCS the GA *does not* influence action selection by preventing some actions from being selected. In XCS, all actions are eligible for selection, just as in tabular Q-learning.[5]

This point is somewhat subtle, since the same action selection method can be used with both systems. The behaviour of the chosen method differs in the two systems because of the form of representation upon which it operates (partial versus complete maps). In complex systems like classifier systems, this sort of interaction between components can easily occur by chance, although in this case XCS's designer, Stewart Wilson, intended action selection to operate on complete maps [43]. What Wilson may not have intended was the resulting decoupling of the GA and action selection, i.e., of the GA and policy determination.

That XCS's GA is detached from action selection seems highly significant to the issue of how it determines policies. It is also relevant to the issue of the GA and RL-views, as we'll see shortly.

4.3 Three Approaches to Determining a Policy

We can distinguish three ways to determine a policy. The first is to evolve and evaluate complete policies, in which case policies are completely determined by the evolutionary algorithm. This is the approach used in Pittsburgh LCS. A second approach is to derive a policy from a (complete) value function, as we do in tabular Q-learning. In this case the policy is completely determined by the Q-learning process. This is the approach used with XCS, where the value function is stored by the rule population.

Finally, the third approach, that used by SBXCS, lies somewhere between the first two. In SBXCS, the policy is derived from the incomplete representation of the value function SBXCS's partial map provides. Significantly, the policies which can be derived depend on which actions are advocated, which in turn depends on which rules the GA maintains in the population. Thus, the policy is

[5] More carefully, initialising XCS with an empty population results in a temporarily partial map, allowing the GA and covering some influence on action selection.

determined by a combination of Q-learning and evolution (and any additional rule discovery mechanisms, such as covering).[6]

Note that the GA in XCS *does* influence the policy to the extent that it influences the value function by aliasing states (introducing #s in rule conditions). But this is less direct than the GA's influence on the policy in SBXCS.

4.4 The GA-View and the RL-View

In this section we relate the two learning subsystems to the two primary tasks (policy learning and generalisation) of the classifier system, consider the emphasis to be placed on the role of the GA, and finally propose an answer to the question of whether classifier systems are GA-based or RL-based.

Which Subsystem Does What? Although all classifier systems contain a rule discovery and credit assignment system, we can now see that different classifier systems operate on very different principles. Although XCS and SBXCS are very similar algorithmically, XCS relies entirely on Q-learning to obtain policies, while SBXCS combines Q-learning with evolution. Pittsburgh classifier systems rely on the GA to address both issues. Figure 3 summarises the differences in the three approaches.

	XCS	**SBXCS**	**Pitt LCS**
Policy	QL	QL & GA	GA
Generalisation	GA	GA	GA

Fig. 3. The method (Genetic Algorithm or Q-Learning) employed by each system for finding policies and generalisation.

The Importance of the GA The two approaches place different emphasis on the role of the GA; in SBXCS the GA is more significant as it directly affects the policy. This emphasis on the role of the GA is part of the rationale for traditional (strength-based) classifier systems. In case there is any doubt concerning the emphasis to be placed on the role of Q-learning in accuracy-based XCS, [18] shows how XCS reduces to tabular Q-learning when generalisation is disabled. The quotations at the start of this work seem entirely consistent with these conclusions regarding the importance of the GA.

[6] The fighter aircraft LCS [26] is an interesting case in which a default policy is applied in the absence of any matching rule.

Which View for Which System? The RL-view seems entirely appropriate for XCS, since it derives its policy solely from a value function, just as tabular Q-learning does. In contrast, the GA-view seems entirely appropriate for Pittsburgh LCS, which rely on the GA for both policy learning and generalisation. Somewhere in between the two extremes lies SBXCS, in which the GA and Q-learning both contribute to determining the policy.

5 Conclusion

I have tried to convey two somewhat imprecise points of view on what a classifier system is, and how it solves problems (adapts its policy). The GA-view holds that classifier systems solve problems using a GA, while the RL-view holds that they solve them as tabular Q-learning does.

An attempt was made to make these views more concrete by considering the goals of a classifier system, and how two particular systems operate to achieve these goals. We have considered how XCS and its strength-based twin SBXCS differ in how they determine their policies, and seen that XCS's GA has far less influence on the policy that SBXCS's. This suggests the GA-view is more appropriate for SBXCS (and other strength-based LCS like it). In contrast, the RL-view seems entirely appropriate for XCS. Indeed, it has been shown elsewhere that XCS is a proper generalisation of tabular Q-learning [18]. These differences indicate that XCS differs greatly from other classifier systems, even its twin SBXCS.

6 Acknowledgements

I am grateful to Stewart Wilson and Alwyn Barry for our our many discussions of XCS, and to Robert Smith for his comments.

References

1. Manu Ahluwalia and Larry Bull. A Genetic Programming-based Classifier System. In Banzhaf et al. [2], pages 11–18.
2. W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors. *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*. Morgan Kaufmann, 1999.
3. Alwyn Barry. *XCS Performance and Population Structure within Multiple-Step Environments*. PhD thesis, Queens University Belfast, 2000.
4. Lashon B. Booker. Improving the performance of genetic algorithms in classifier systems. In John J. Grefenstette, editor, *Proceedings of the 1st International Conference on Genetic Algorithms and their Applications (ICGA-85)*, pages 80–92, Pittsburgh, PA, July 1985. Lawrence Erlbaum Associates.
5. Martin Butz, David E. Goldberg, and Wolfgang Stolzmann. New challenges for an ACS: Hard problems and possible solutions. Technical Report 99019, University of Illinois at Urbana-Champaign, Urbana, IL, October 1999.

6. Martin Butz and Wolfgang Stolzmann. Action-Planning in Anticipatory Classifier Systems. In Wu [46], pages 242–249.
7. Martin V. Butz, David E. Goldberg, and Wolfgang Stolzmann. Probability-enhanced predictions in the anticipatory classifier system. In *Proceedings of the International Workshop on Learning Classifier Systems (IWLCS-2000), in the Joint Workshops of SAB 2000 and PPSN 2000*, 2000.
8. Martin V. Butz and Stewart W. Wilson. An Algorithmic Description of XCS. In Pier Luca Lanzi, Wolfgang Stolzmann, and Stewart W. Wilson, editors, *Advances in Learning Classifier Systems*, number 1996 in LNAI, pages 253–272. Springer–Verlag, 2001.
9. Henry Brown Cribbs III and Robert E. Smith. What Can I do with a Learning Classifier System? In C. Karr and L. M. Freeman, editors, *Industrial Applications of Genetic Algorithms*, pages 299–320. CRC Press, 1998.
10. Attilio Giordana and Filippo Neri. Search-Intensive Concept Induction. *Evolutionary Computation*, 3:375–416, 1995.
11. David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Reading, MA, 1989.
12. John J. Grefenstette. Credit Assignment in Rule Discovery Systems Based on Genetic Algorithms. *Machine Learning*, 3:225–245, 1988.
13. John H. Holland. Adaptation. In R. Rosen and F. M. Snell, editors, *Progress in theoretical biology.* New York: Plenum, 1976.
14. John H. Holland. Properties of the bucket brigade. In John J. Grefenstette, editor, *Proceedings of the 1st International Conference on Genetic Algorithms and their Applications*, pages 1–7, Pittsburgh, PA, July 1985. Lawrence Erlbaum Associates.
15. John H. Holland, Lashon B. Booker, Marco Colombetti, Marco Dorigo, David E. Goldberg, Stephanie Forrest, Rick L. Riolo, Robert E. Smith, Pier Luca Lanzi, Wolfgang Stolzmann, and Stewart W. Wilson. What is a Learning Classifier System? In Lanzi et al. [20], pages 3–32.
16. John H. Holland and J. S. Reitman. Cognitive systems based on adaptive algorithms. In D. A. Waterman and F. Hayes-Roth, editors, *Pattern-directed inference systems.* New York: Academic Press, 1978. Reprinted in: Evolutionary Computation. The Fossil Record. David B. Fogel (Ed.) IEEE Press, 1998. ISBN: 0-7803-3481-7.
17. Hisao Ishibuchi and Tomoharu Nakashima. Linguistic Rule Extraction by Genetics-Based Machine Learning. In Darrell Whitely, David Goldberg, Erick Cantú-Paz, Lee Spector, Ian Parmee, and Hans-Georg Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, pages 195–202. Morgan Kaufmann, 2000.
18. Tim Kovacs. *A Comparison of Strength and Accuracy-Based Fitness in Learning Classifier Systems.* PhD thesis, School of Computer Science, University of Birmingham, 2001.
19. Pier Luca Lanzi. Extending the Representation of Classifier Conditions Part II: From Messy Coding to S-Expressions. In Banzhaf et al. [2], pages 345–352.
20. Pier Luca Lanzi, Wolfgang Stolzmann, and Stewart W. Wilson, editors. *Learning Classifier Systems. From Foundations to Applications*, volume 1813 of *LNAI*. Springer-Verlag, Berlin, 2000.
21. Rick L. Riolo. Bucket Brigade Performance: I. Long Sequences of Classifiers. In *Proceedings Second International Conference on Genetic Algorithms (ICGA-87)*, pages 184–195. Lawrence Erlbaum Associates, 1987.
22. Rick L. Riolo. The emergence of coupled sequences of classifiers. In Schaffer [23], pages 256–264.

23. J. David Schaffer, editor. *Proceedings of the 3rd International Conference on Genetic Algorithms (ICGA-89)*, George Mason University, June 1989. Morgan Kaufmann.
24. Robert E. Smith. A Report on The First International Workshop on Learning Classifier Systems (IWLCS-92). NASA Johnson Space Center, Houston, Texas, Oct. 6-9, 1992.
25. Robert E. Smith. Derivative Methods: Learning Classifier Systems. In Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors, *Handbook of Evolutionary Computation*, pages B1.2:6–B1.5:11. Institute of Physics Publishing and Oxford University Press, 1997. http://www.iop.org/Books/Catalogue/
26. Robert E. Smith, B. A. Dike, B. Ravichandran, A. El-Fallah, and R. K. Mehra. The Fighter Aircraft LCS: A Case of Different LCS Goals and Techniques. In Lanzi et al. [20], pages 283–300.
27. S. F. Smith. *A Learning System Based on Genetic Adaptive Algorithms.* PhD thesis, University of Pittsburgh, 1980.
28. S. F. Smith. Flexible Learning of Problem Solving Heuristics through Adaptive Search. In *Proceedings Eight International Joint Conference on Artificial Intelligence*, pages 422–425, 1983.
29. S. F. Smith. Adaptive learning systems. In R. Forsyth, editor, *Expert systems: Principles and case studies*, pages 169–189. Chapman and Hall, 1984.
30. Wolfgang Stolzmann. Learning classifier systems using the cognitive mechanism of anticipatory behavioral control, detailed version. In *Proceedings of the First European Workshop on Cognitive Modelling*, pages 82–89. Berlin: TU, 1996.
31. Wolfgang Stolzmann. *Antizipative Classifier Systeme.* PhD thesis, Fachbereich Mathematik/Informatik, University of Osnabrück, 1997.
32. Wolfgang Stolzmann. Two Applications of Anticipatory Classifier Systems (ACSs). In *Proceedings of the 2nd European Conference on Cognitive Science*, pages 68–73. Manchester, U.K., 1997.
33. Wolfgang Stolzmann. Anticipatory classifier systems. In *Proceedings of the Third Annual Genetic Programming Conference*, pages 658–664. Morgan Kaufmann, 1998.
34. Wolfgang Stolzmann. Latent Learning in Khepera Robots with Anticipatory Classifier Systems. In Wu [46], pages 290–297.
35. Wolfgang Stolzmann. An Introduction to Anticipatory Classifier Systems. In Lanzi et al. [20], pages 175–194.
36. Wolfgang Stolzmann and Martin Butz. Latent Learning and Action-Planning in Robots with Anticipatory Classifier Systems. In Lanzi et al. [20], pages 301–317.
37. Wolfgang Stolzmann, Martin Butz, J. Hoffmann, and D. E. Goldberg. First cognitive capabilities in the anticipatory classifier system. In J. A. Meyer et al., editor, *From Animals to Animats 6: Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior*, pages 287–296, 2000. Also Technical Report 2000008 of the Illinois Genetic Algorithms Laboratory.
38. Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, MA, 1998.
39. Andy Tomlinson and Larry Bull. A Corporate Classifier System. In A. E. Eiben, T. Bäck, M. Shoenauer, and H.-P. Schwefel, editors, *Proceedings of the Fifth International Conference on Parallel Problem Solving From Nature – PPSN V*, volume 1498 of *Lecture Notes in Computer Science*, pages 550–559. Springer Verlag, 1998.
40. Patrick Tufts. Evolution of a Clustering Scheme for Classifier Systems: Beyond the Bucket Brigade. PhD Thesis proposal, 1994.

41. Patrick Tufts. Dynamic Classifiers: Genetic Programming and Classifier Systems. In E. V. Siegel and J. R. Koza, editors, *Working Notes for the AAAI Symposium on Genetic Programming*, pages 114–119, MIT, Cambridge, MA, USA, 1995. AAAI.
42. Stewart W. Wilson. ZCS: A Zeroth Level Classifier System. *Evolutionary Computation*, 2(1):1–18, 1994.
43. Stewart W. Wilson. Classifier Fitness Based on Accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.
44. Stewart W. Wilson. Personal communication to Alwyn Barry and Tim Kovacs. June 12, 1998.
45. Stewart W. Wilson and David E. Goldberg. A Critical Review of Classifier Systems. In Schaffer [23], pages 244–255.
46. Annie S. Wu, editor. *Proceedings of the 1999 Genetic and Evolutionary Computation Conference Workshop Program*, 1999.