Investigating Generalization in the Anticipatory Classifier System

Martin V. Butz[1], David E. Goldberg[1], and Wolfgang Stolzmann[2]

[1] Illinois Genetic Algorithms Laboratory
University of Illinois at Urbana-Champaign, IL, USA,
{butz, deg}@illgal.ge.uiuc.edu
[2] Institut für Psychologie III
Universität Würzburg, Germany
stolzmann@psychologie.uni-wuerzburg.de

Abstract. Recently, a genetic algorithm (GA) was introduced to the Anticipatory Classifier System (ACS) which surmounted the occasional problem of over-specialization of rules. This paper investigates the resulting generalization capabilities further by monitoring the performance of the ACS in the highly challenging multiplexer task in detail. Moreover, by comparing the ACS to the XCS classifier system in this task it is shown that the ACS generates accurate, maximally general rules and its population converges to those rules. Besides the observed ability of latent learning and the formation of an internal environmental representation, this ability of generalization adds a new advantage to the ACS in comparison with similar approaches.

1 Introduction

Generalization is one of the most important issues to enable problem solving in nature. Higher animals filter out which information is important to satisfy each of their needs. Instincts can be viewed as evolved generalized rules dependent on only a few stimuli. In artificial systems generalization is as important as in nature. For example an autonomous robot equipped with many different sensors must be able to generalize to solve similar tasks. Generalization is a powerful property of learning classifier systems (LCSs).

The Anticipatory Classifier System (ACS) (Stolzmann, 1998) enhances the rule structure of traditional LCSs with an effect part anticipating the consequences of an action in a certain situation. This additional effect part opens the portal to an internal representation of the perceived environment and thus to many cognitive processes such as lookahead planning (Stolzmann & Butz, 2000), mental acting and lookahead action selection (Stolzmann, Butz, Hoffmann, & Goldberg, 2000). While the great benefits of the cognitive capabilities were shown in the cited papers, the aim herein is to analyze the genetic generalization capabilities, introduced in Butz, Goldberg, and Stolzmann (2000a), further using the multiplexer test bed. Although this problem is a one step problem and thus not typical for the ACS, it serves well to analyze the generalization capabilities of the system. The test bed was already successfully applied in the ACS (Butz, Goldberg, & Stolzmann, 2000b) but a complete convergence could not be shown. This paper will show that the ACS is able to converge to the accurate, maximally general rules. Furthermore, the performance will be compared to the XCS classifier system (Wilson, 1995), which is able to solve the multiplexer problem with an optimal generalization (Wilson, 1998).

The next section will give an overview of the ACS with the present specialization and generalization pressure and the interaction of the two pressures. After that, Section 3 introduces the multiplexer environment and analyzes the performance of the ACS. Finally, a summary is given.

2 Overview of the Anticipatory Classifier System

The ACS is based on LCSs and on the psychological learning theory of anticipatory behavioral control (Hoffmann, 1993) that is realized by the anticipatory learning process (ALP). The theory states that learning is directed by the need to correctly anticipate behavioral consequences; thus requiring the construction of an internal representation of the environment. The enhanced LCS structure combined with the ALP realizes this need in the ACS.

However, the ACS is not the first mechanism that is able to form an internal environmental representation. Sutton (1991) introduced the Dyna-Q algorithm which uses an environmental representation to speed up the reinforcement learning capabilities. Holland (1990) and Riolo (1991) developed another classifier system which was able to form an environmental representation. Drescher (1991) published a schemata approach that is based on the Piagetian development theory. The ACS differs from these approaches in its directed rule discovery due to the realization of the anticipatory behavioral control theory as well as the type of genetic generalization.

2.1 The Basics of the ACS

The ACS perceives states $\sigma \in \{s_1, s_2, ..., s_m\}^L$ from the environment, where L is the length of the strings, m is the number of possible different values, and $s_1, ..., s_m$ are the different possible values. It acts upon the environment with possible actions α that are also coded with discrete values $\alpha \in \{r_1, r_2, ..., r_n\}$, where n is the number of different actions and $r_1, ..., r_n$ are the different actions. After the execution of an action it receives a payoff ρ, a simple real value.

The knowledge of the ACS is represented by its population of classifiers. Each classifier has a condition part C, an action part A, an effect part E, a mark M, a time stamp t_s, a numerosity num, a quality measure q, and a reward prediction measure r. C and E have the structure $\{s_1, ..., s_m, \#\}^L$. A '#'-symbol in an attribute of C, which is called 'don't-care' symbol, matches any perceived value in this attribute. The more 'don't-care' symbols are in C, the more general a classifier is. A '#'-symbol in an attribute of E, which is called 'pass-through' symbol, predicts that this attribute will remain the same (i.e. the corresponding value in the perception does not change after the execution of an action). The action part A consists out of any action which is possible in the environment (i.e. $A \in \alpha$). The mark has the structure $M = (m_1, ..., m_L)$ with $m_i \subseteq \{s_1, ..., s_m\}$. M serves as a remainder that the classifier did not work correctly sometimes as well as specifies the situations where it did not work correctly. Due to its structure it is assured that its size does not exceed $L * m$. num gives the classifier the property of a macro-classifier identical to XCS (Wilson, 1995). Each classifier can represent several identical classifiers in the population. This representation is used for an increase in the speed of the system and affects the performance only slightly. q measures the quality of the anticipations of the classifier and r predicts the reward.

At time t, a behavioral act starts with the perception of the current situation $\sigma(t)$. First, the ACS forms a match set out of all classifiers whose conditions match the current situation. Next, the ACS chooses an action (here randomly), executes the action, and forms an action set out of the match set. Using the perceptions of the next state $\sigma(t+1)$ it forms a learning set out of the action set by applying the ALP. The ALP realizes the latent learning and the formation of the environmental representation. It resembles the specialization pressure in the ACS. Finally, the actual payoff $\rho(t)$ combined with the predicted payoff in the resulting situation is distributed among the classifiers in the resulting learning set applying an update similar to Q-learning with discount factor γ and learning rate b_r:

$$r = (1 - b_r) * r + b_r * (\gamma * max_{cl \in M_{set}(t+1)}(q_{cl} * r_{cl}) + \rho(t)) \quad (1)$$

2.2 The Specialization Pressure

The ACS assumes in the beginning that no action has any effect. (i.e. each possible action is represented by one classifier that only has '#'-symbols in C and E). The ALP then carefully generates more specific rules out of those most general ones. It is applied in the action set and distinguishes the following cases:

- A classifier anticipates $\sigma(t+1)$ incorrectly. In this case the incorrect classifier is marked with $\sigma(t)$ and its quality is decreased. Furthermore:
  - If E can be further specialized (i.e. '#'-symbols can be changed to specific values) to correct the anticipations, then a new classifier is built which is specified in these attributes in C and E.
  - If E cannot be further specialized or nothing changed in the environment at all, then no new classifier is created.
- A classifier anticipates $\sigma(t+1)$ correctly. Now the quality is increased. Furthermore, if the classifier is marked (which means that the classifier anticipated incorrectly in earlier experienced situations) and it can be further specialized, a new classifier is formed which is more specialized in C. It can be further specialized, if it is not already completely specialized and if the number of specified unchanging attributes (i.e. attributes which have non-'#'symbols in C but '#'-symbols in E) does not exceed the threshold u_{max}.

An increase or decrease of q is realized by the Widrow-Hoff delta rule with learning rate b_q (i.e. $q = (1 - b_q) * q + b_q$ or $q = (1 - b_q) * q$). If q decreases under $\theta_i = 0.1$ the classifier is deleted. If q increases over $\theta_r = 0.9$ the classifier is considered as *reliable* and thus part of the internal environmental representation. If a generated classifier already exists, the newly generated classifier is not inserted into the population but the quality of the old one is increased.

Although the ALP generates more specialized classifiers carefully there are some causes for over-specialization. (1) The ALP specializes the changing attributes in a classifier always in C and E although the specialization in C is not always necessary. (2) The information in the mark can be misleading due to its limited structure and due to the possible only partial observation of the environment. (3) Non deterministic environments can further mislead the ALP.

2.3 The Genetic Generalization Pressure

The causes of over-specialization are overcome now with a genetic algorithm that is biased towards generalization. The GA works as follows:

- The GA is applied in the action set realizing a niche GA.
- In order to control the frequency of the GA application, the GA is applied in an action set only if the average time since the last GA application in the set is greater than the threshold θ_{GA}.
- Two classifiers are selected by roulette wheel selection using q^3 as the bid for each classifier.
- The parents remain in the population.
- Each specific attribute in the conditions is changed to a '#'-symbol with a probability μ.
- Crossover is applied with a probability χ.
- The resulting classifiers are only inserted into the population if they are not completely general.
- If the action set size exceeds the action set size threshold θ_{as} as many classifiers are deleted in the action set as necessary.
- The deletion method uses tournament selection with sizes of 1/3 of the action set size. The classifier with the lowest quality is deleted. If two classifiers have approximately the same quality q (i.e. $|q_1 - q_2| < 0.1$), then the one with more specific attributes in C is deleted. Thus, the deletion method is a niche deletion which favors formal generality.

If a newly generated classifier already exists, it is not directly inserted but the numerosity *num* of the old classifier is increased consequently representing the new one as well.

The generalization pressure is realized by the (1) biased selection which chooses high quality classifiers that consequently must be at least specific enough, by the (2) generalizing mutation which only generates more general offspring, and by the (3) biased deletion which prefers inaccurate but also over-specialized classifiers.

2.4 The Interaction of the ALP and the GA

While the ALP specializes and sometimes over-specializes, the GA fixes the over-specializations but also sometimes over-generalizes. In order to keep the two pressures in equilibrium we use subsumption deletion similar to XCS (Wilson, 1998). When a new classifier is generated, the ACS checks if a classifier exists in the population that subsumes the new classifier. A classifier subsumes a new classifier if it is (1) reliable, (2) experienced (i.e. the classifier was applied more than 20 times), (3) not marked, (4) more general, and (5) has the same action. If such a classifier exists, then the new classifier is not added to the population. Furthermore, q or num of the old classifier is increased when the ALP or the GA created the new one, respectively. Thus, the ALP and the GA does not create over-specialized classifiers once the maximally general classifiers are found. However, the GA still creates over-general classifiers out of the maximally general ones. To prevent the over-generalization due to the GA, the specialization pressure needs to be strong enough. As we will see in 3.1, the specialization pressure can be influenced by the specificity threshold u_{max} (see Section 2.2).

3 The Multiplexer Problem

We chose the multiplexer problem, a one-step problem, to investigate the generalization capabilities in the ACS. In order to make the problem solvable we need to code a perceptual causality. This is done by coding the reward into the perceived consequences. The task is especially hard as it mainly asks for a further differentiation of the conditions but learning in the ACS is primarily based on the correct anticipation of behavioral consequences (see Section 2).

The multiplexer function is defined for binary strings of length $l = k + 2^k$. Its value is determined by the address specified by the first k bits in the remaining 2^k bits. The value of the string '100010' in the 6-Multiplexer function is 1, since the first two bits address the third bit in the remaining four bits. The function can also be written in disjunctive normal form: $F_6 = \neg x_1 \neg x_2 x_3 \vee \neg x_1 x_2 x_4 \vee x_1 \neg x_2 x_5 \vee x_1 x_2 x_6$. Thus, in the multiplexer problem there are 2^{k+1} constellations to learn which can be classified as either correct or wrong.

As mentioned above, we need to code the reward into the perceptions in order to supply a perceptual causality. Wilson (1995) published a reward map for the multiplexer problem which divides the payoff distinguishing between the 2^k possible constellations: Payoff $= 300 + 200 * (int) first\_k\_bits + 100 * (addressed\_bit)$ for correct classifications and $200 * (int) first\_k\_bits + 100 * (addressed\_bit)$ for incorrect ones. Later, Wilson (1998) also solved the multiplexer problem with a payoff distinguishing only between correct and wrong classifications (payoff 1000 and 0). Either variation may be coded into the perceptions of the ACS. When coding the reward map, there are $k + 2$ additional bits which are always 0 in a given situation and change in the effects according to the reward map. In order to supply always a change in the perceptions we start with one for the lowest payoff possible instead of zero. When coding the payoff 1000/0, then only one additional attribute is present that changes to one or to two when the classification is wrong or correct, respectively.

One problem instance starts with the presentation of a random bit string of length $k + 2^k$ plus the additional attributes which are set to zero. Next, the ACS chooses an action randomly (i.e. the classification) and the result is coded into the perceptions, leaving the random bit string unchanged. After the ACS learned from the consequences, one problem ends and a new random instance is presented.

3.1 Trouble with Over-Specialization

The 11-Multiplexer problem was already investigated in Butz, Goldberg, and Stolzmann (2000b). Although the ACS solved the problem with a number of explored problems comparable to XCS, the generalization was not as good as in XCS. The population increased to a higher level and converged not as fast as the population in XCS. Further investigations showed that the higher population size was mainly caused by over-specialization at the beginning of the run. Because of the restriction in the structure of the mark the ACS is not able to find out which components are the most relevant ones to specialize and thus specializes all components at once. Since the GA is a niche GA and only works in specific action sets, these over-specialized classifiers hide in the population. They rarely match the current state and are therefore very seldom part of the GA. Thus, they stay in the population, although a correct, more general classifier was found long ago, which could subsume the over-specialized one(s).

There are a number of possibilities to fix this problem. One possibility is a population sweep where the accurate, more general classifiers absorb the over-specialized ones. Another possibility is to restrict the population size and execute a deletion in the population which preferably deletes over-specialized classifiers. These are global solutions tackling the problem from the top level. In this paper however, we approach the problem from the other side and consequently restrict the over-specialization. For doing that, the ACS has got the threshold u_{max}, the maximal number of specialized attributes in the conditions that are predicted not to change. Restricting the number of attributes to specialize, the ACS will not generate over-specialized classifiers anymore and the population should converge further. This latter approach is applied herein.

Figure 1 shows the resulting performance of the ACS in the 11- and the previously unsolved 20-Multiplexer problem. The 'correct anticipations' measure is calculated by presenting a random instance of the problem, choosing the most probable classifiers for both classifications (i.e. the classifiers with the highest q), and monitoring if the effects were anticipated correctly by the classifiers. All results are averaged over ten runs. The parameter settings in this and the following experiments were $b_q = b_r = 0.05$, $\theta_{GA} = 25$, $\chi = 0.0$, $\mu = 0.3$, $\theta_{as} = 20$. Experiments with crossover did not influence the performance in this problem.

In both graphs we can observe how the population size and the knowledge depend on the constant u_{max}. By limiting the number of specified unchanging attributes, the specialization pressure is decreased. When choosing a low threshold, the population stays small, but a complete knowledge is not necessarily reached because the GA over-generalizes and the ALP is too slow to recover. When choosing a higher threshold, the population size increases and the knowledge reaches 100%. However, when u_{max} is chosen too high, the convergence takes longer and complete knowledge is achieved more slowly since the ALP over-specializes and the GA is not able to generalize fast enough.

3.2 The Two Codings

The ACS is based on learning from anticipations. Thus, the more effects that distinguish a problem, the easier it should be to learn a problem for the ACS.

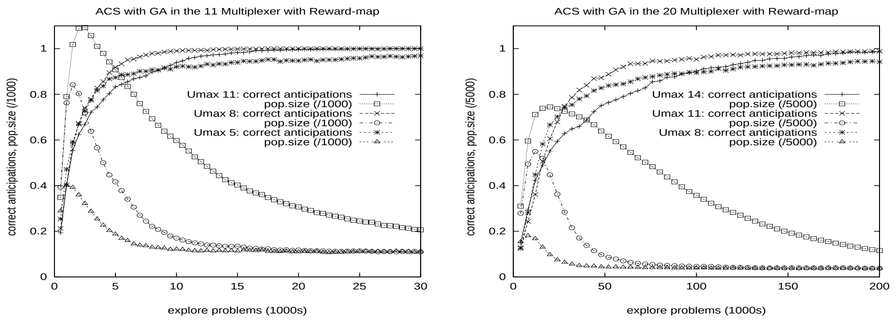

Fig. 1. The comparison of different u_{max} values reveals the dependency of the size of the evolving population and the quality of the evolving knowledge.

Figure 2 shows exactly this in the 11- and 20-Multiplexer. When only one additional attribute is coded into the perceptions distinguishing between correct and wrong classification (i.e. the 1/2 coding), the ACS takes longer to evolve a complete knowledge and the population required is larger. When the whole reward map is coded into the perceptions, the different effects result in a faster partitioning of the population and thus enable the ACS to learn more quickly.

Moreover, it can be observed that the knowledge in the 1/2 coded problem does not reach 100% knowledge completely. Analyzing the problem revealed that sometimes 'parasitic' classifiers evolve. These classifiers are correct and maximally general but occupy half of the space of each of two good classifiers. Considering the 6-Multiplexer problem such a classifier would be '1###11-1' (showing condition – action). The classifier occupies half of the space of classifier '11###1-1' and of classifier '10##1#-1'. Once a parasitic classifier took over a niche the good classifiers get lost, and thus, half of their covered space is suddenly empty. This problem is currently unsolved and needs further investigation.

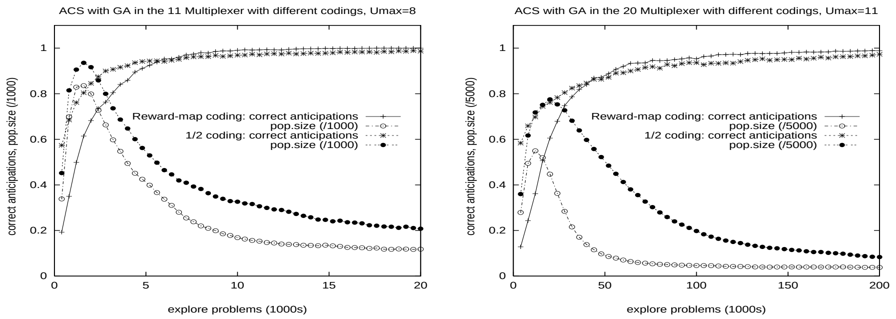

Fig. 2. The comparison of different codings shows that the ACS is able to increase performance when the complete reward map is coded into the perceptions.

However, the graphs show that the effect is minor and that the ACS is able to recover from this error, since the knowledge still reaches a very high level.

3.3 Comparing the ACS with XCS

In order to investigate the quality of the ACS performance further, this section gives a direct comparison to XCS. We emphasize that the comparison is one of apple and oranges. The ACS is basically learning C-R-E relations and thus an internal environmental representation, whereas XCS is learning C-R-payoff relations and thus a payoff-map of the environment. This difference results normally in a different area of application. However, the multiplexer problem helps to investigate generalization in both approaches.

The XCS results presented here are run with an XCS implementation which is based on the original Wilson (1995) paper. The GA is executed in the action set and subsumption deletion is used according to Wilson (1998). The constants are chosen as in the two mentioned papers. Similar to the XCS performance measure we use a payoff based performance measure for the ACS in the subsequent results. After each 'explore trial' one 'exploit trial' is executed. In this 'exploit trial' the better action is executed which is determined by a prediction array identical to XCS. In the prediction array, the reward prediction of each possible action is determined according to the quality q and the reward prediction r of each matching classifier with the according action:

$$Value(a) = \sum_{cl \in M_{set} \wedge A_{cl}=a} (r_{cl} * q_{cl} * num_{cl}) / \sum_{cl \in M_{set} \wedge A_{cl}=a} q_{cl} * num_{cl} \qquad (2)$$

Since the learning mechanism in the ACS is based on the learning of effects we always give a payoff of 1000/0 to the ACS regardless of which coding is used. Figure 3 compares the performance in the reward map multiplexer task and the 1000/0 payoff learning task. It can be observed that both algorithms benefit from the reward map. Moreover, although the size of the population of the ACS increases to a slightly higher level than XCS in the beginning, it converges to a comparable level in all four tasks. The performance in classifying is similar to XCS. While the ACS beats the XCS performance in the beginning of a run, it takes the ACS slightly longer to converge to a 100% performance level. However, the convergence of the population size is very similar to the XCS convergence. This shows that the ACS is generating the desired accurate, maximally general classifiers.

4 Summary

This paper has studied the generalization capabilities of the ACS with a GA in the highly challenging multiplexer problem. Even when only one additional attribute created the perceptual causality, the ACS was able to scale up in the 11- and 20-Multiplexer task. Monitoring the reward learning, the performance was very similar to XCS's performance. Thus, transferring Wilson's conclusions to his generalization hypothesis (Wilson, 1995) and the investigations about the refinement to an optimality hypothesis (Kovacs, 1996), the ACS also generates accurate, maximally general classifiers and the population is converging

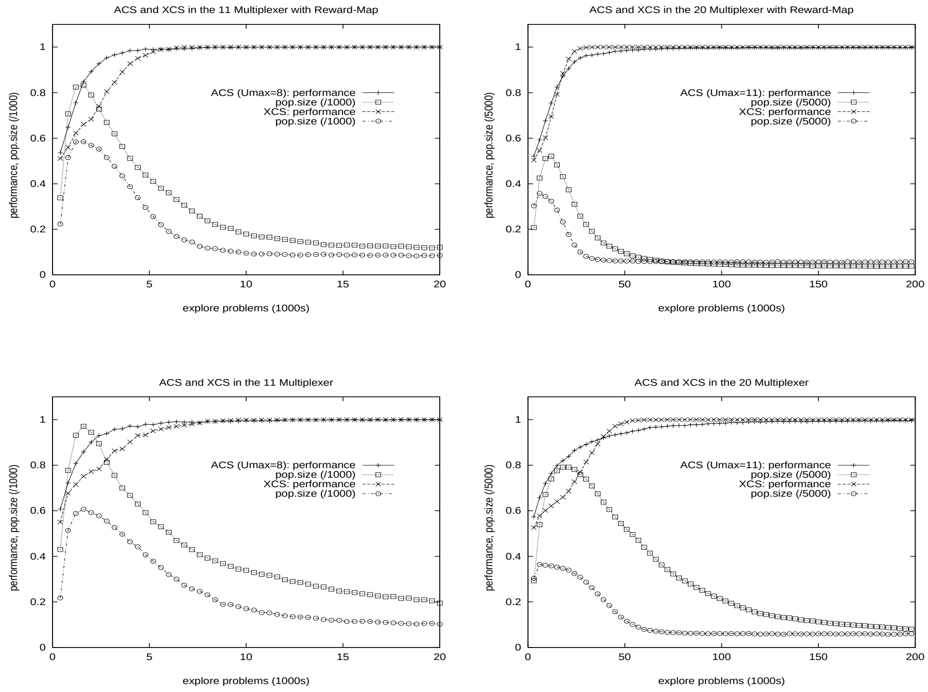

Fig. 3. The ACS and XCS in the multiplexer problem learning the reward-map and the 1000/0 payoff.

to those classifiers. However, the GA is not working on its own but the process is a combination of a directed specialization due to the ALP and an indirected generalization due to the GA. Furthermore, the learned model is not a condition-action-payoff model but a condition-action-effect model of an environment. This *generalized* internal model gives the ACS a great advantage in comparison with all other approaches that form internal environmental representations.

5 Acknowledgments

The authors would like to thank Martin Pelikan and Stewart Wilson for valuable discussions and useful comments as well as the Automated Learning Group of NCSA at the University of Illinois in Urbana-Champaign for their support. The work was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grants F49620-94-1-0103, F49620-95-1-0338, F49620-97-1-0050, and F49620-00-0163. Research funding for this work was also provided by a grant from the National Science Foundation under grant DMI-9908252. Support was also provided by a grant from the U. S. Army Research Laboratory under the Federated Laboratory Program, Cooperative Agreement DAAL01-96-2-0003. The US Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon. Additionally, research funding was provided by the

German Research Foundation DFG. The views and conclusions contained herein are my own and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research, the National Science Foundation, the U. S. Army, or the U.S. Government.

References

[Butz, Goldberg, & Stolzmann, 2000a] Butz, M. V., Goldberg, D. E., & Stolzmann, W. (2000a). Introducing a genetic generalization pressure to the anticipatory classifier system - Part 1: Theoretical approach. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*. In press.

[Butz, Goldberg, & Stolzmann, 2000b] Butz, M. V., Goldberg, D. E., & Stolzmann, W. (2000b). Introducing a genetic generalization pressure to the anticipatory classifier system - Part 2: Performance analysis. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*. In press.

[Drescher, 1991] Drescher, G. L. (1991). *Made-Up Minds, a constructivist approach to artificial intelligence*. Cambridge, MA: MIT Press.

[Hoffmann, 1993] Hoffmann, J. (1993). *Vorhersage und Erkenntnis [Anticipation and Cognition]*. Goettingen, Germany: Hogrefe.

[Holland, 1990] Holland, J. (1990). Concerning the emergence of tag-mediated lookahead in classifier systems. *Special issue of Physica D*, *42*, 188–201.

[Kovacs, 1996] Kovacs, T. (1996). *Evolving Optimal Populations with XCS Classifier Systems*. Master's thesis, School of Computer Science, University of Birmingham, Birmingham, U.K. Also tech. report CSR-96-17 and CSRP-96-17 ftp://ftp.cs.bham.ac.uk/pub/tech-reports/1996/CSRP-96-17.ps.gz.

[Riolo, 1991] Riolo, R. L. (1991). Lookahead planning and latent learning in a classifier system. In Meyer, J.-A., & Wilson, S. W. (Eds.), *Proceedings of the first international conference on simulation of adaptive behavior* pp. 316–326. Cambridge, MA: MIT Press.

[Stolzmann, 1998] Stolzmann, W. (1998). Anticipatory Classifier Systems. In *Genetic Programming '98* pp. 658–664. University of Wisconsin, Madison, Wisconsin: Morgan Kaufmann.

[Stolzmann & Butz, 2000] Stolzmann, W., & Butz, M. V. (2000). Latent learning and action-planning in robots with Anticipatory Classifier Systems. In Lanzi, P. L., Stolzmann, W., & Wilson, S. W. (Eds.), *Learning Classifier Systems: An Introduction to Contemporary Research, LNAI 1813* Berlin Heidelberg: Springer-Verlag.

[Stolzmann, Butz, Hoffmann, & Goldberg, 2000] Stolzmann, W., Butz, M. V., Hoffmann, J., & Goldberg, D. E. (2000). First cognitive capabilities in the anticipatory classifier system. *Proceedings of the sixth international conference on the Simulation of Adaptive Behavior (SAB2000)*. In press.

[Sutton, 1991] Sutton, R. S. (1991). Reinforcement learning architectures for animats. In Meyer, J.-A., & Wilson, S. W. (Eds.), *Proceedings of the first international conference on simulation of adaptive behavior* pp. 288–296. Cambridge, MA: MIT Press.

[Wilson, 1995] Wilson, S. W. (1995). Classifier fitness based on accuracy. *Evolutionary Computation*, *3*(2), 149–175.

[Wilson, 1998] Wilson, S. W. (1998). Generalization in the XCS classifier system. In Koza, J. R. e. a. (Ed.), *Genetic Programming 1998: Proceedings of the third annual conference* pp. 665–674. San Francisco: Morgan Kaufmann.