The Anticipatory Classifier System and Genetic Generalization

A. Martin V. Butz (butz@psychologie.uni-wuerzburg.de)
Department of Cognitive Psychology
University of Würzburg
Würzburg, Germany

B. David E. Goldberg (deg@illigal.ge.uiuc.edu)
Illinois Genetic Algorithms Laboratory
University of Illinois at Urbana-Champaign
Illinois, USA

C. Wolfgang Stolzmann
(wolfgang.stolzmann@daimlerchrysler.com)
DaimlerChrysler AG
Research and Technology
Berlin, Germany

Abstract. The anticipatory classifier system (ACS) combines the learning classifier system framework with the cognitive learning theory of anticipatory behavioral control. The result is an evolutionary system that builds a complete and generalized predictive environmental model. Reinforcement learning techniques are applied to form a behavioral policy represented in the model. After providing some background as well as outlining the objectives of the system, we explain in detail all involved current processes. Furthermore, we analyze the deficiency of over-specialization in the anticipatory learning process (ALP), the main learning mechanism in the ACS. Consequently, we introduce a genetic algorithm (GA) to the ACS that is meant for generalization of over-specialized classifiers. We show that it is possible to form a symbiosis between a directed specialization and a genetic generalization mechanism achieving a learning mechanism that evolves a complete, accurate, and compact description of the perceived environment. Results in three different environmental settings confirm the usefulness of the genetic algorithm in the ACS. Finally, we discuss future research directions with the ACS and anticipatory systems in general.

Keywords: anticipatory learning, anticipatory behavior, genetic generalization, learning classifier systems, online model learning, reinforcement learning

Introduction

The belief in the crucial role of anticipations (i.e. the expectation of action-consequences) in animal learning reaches back to the 19th century (see e.g. James, 1890). Anticipations have a major effect on many organisms altering learning capabilities, behavior, and possibly many other important characteristics. However, only recently has the concept of anticipations acquired more recognition in artificial learning systems.

The question of when it is necessary to include anticipations, and how they should be included, is an important current concern.

Inside the world of artificial and evolutionary learning systems, the recent introduction of accuracy-based fitness in the XCS classifier system (Wilson, 1995) led to a revival of learning classifier systems (LCSs). The accuracy-based approach in XCS overcomes the previously encountered problems in LCSs where especially deferred reward often leads to over-generalization and an unequal distribution of classifiers in the problem space (Wilson and Goldberg, 1989). Moreover, XCS solves the problem of speciation and what mating restriction schemes should be used implicitly by the accuracy measure combined with a niche genetic algorithm (GA).

The accuracy idea and the learning classifier system framework combined with a representation of anticipations constitute the foundation of the anticipatory classifier system (ACS) (Stolzmann, 1997, Stolzmann, 1998). Although the ACS applies a reinforcement learning mechanism, in contrast to other LCSs, the main learning mechanism is not based on reward learning but on anticipatory learning (i.e. learning from the differences between the anticipated and the real perceptual result of an action), the so called latent learning in cognitive sciences. Thus, at this time, ACS is mainly an online generalizing model learner. In order to anticipate consecutive situations, each rule (classifier) in the ACS has an (additional) anticipatory part. An anticipatory learning process (ALP) is applied that intends to build a complete, accurate, and compact model of the perceived environment represented by a population of classifiers. The evolving internal model enables the ACS to use various kinds of cognitive mechanisms such as lookahead planning and mental acting as shown elsewhere (Stolzmann et al., 2000). The ALP mechanism only specializes classifiers and, despite the care of the mechanism, over-specialization occurs sometimes. Thus, the evolving model is not necessarily as compact as possible. The paper solves the problem by applying a genetic generalization mechanism prevailing the over-specialization cases.

The purpose of this article is threefold: First, we place the ACS in a broader picture and provide a comprehensive introduction to the system. Second, we introduce a genetic generalization mechanism to overcome the possible over-specialization defect. Third, we provide empirical results to confirm the resulting capabilities.

The next section provides background information about related systems, the idea and possible necessity of anticipations, and the origin of the ACS. Next, we give a detailed description of the ACS, investigate the causes for over-specialization, and introduce the GA to the ACS and explain the interaction of the GA and the anticipatory learning process.

We then compare and investigate the performance of the ACS without and with GA in a simple gripper scenario, in a multiplexer environment, and two maze environments. Future challenges are discussed after that. Finally, summary and conclusions of the work are provided.

Background

Before focusing on the precise description of the ACS, this section reveals motivations, provides background and origin of the ACS, and discusses related approaches. Two main components are combined in the ACS: the anticipatory approach and the learning classifier system framework. The presence and usage of anticipations in higher animals is the foundation of the anticipatory approach and is reviewed first. Next, we discuss related approaches in mathematics, computer science, and artificial intelligence. Finally, we reveal the LCS motivation and previous model-building approaches in this framework.

The Idea of Anticipations

The insight in psychology that higher animals form an internal environmental representation and adapt their behavior by forming anticipations exploiting the representation reaches far back in history. James gives one of the first hints of the necessity of anticipations. In terms of goal-oriented behavior he states that

> an anticipatory image [...] is the only psychic state which introspection lets us discern as the forerunner of our voluntary acts. (James, 1890, p.1112)

Tolman (1932) proposed that learning is the process of discovering what leads to what. We learn what events lead to other events. According to him, animals learn what is out there to the left and to the right and consequently develop a 'cognitive map'—a sort of internal predictive representation—of the world. Evidence for further learning capabilities beyond the behaviorist stimulus-response learning was firstly manifested in latent learning experiments. For example, experiments showed that rats are indeed learning an internal representation of an environment, without receiving any type of reward. Thislethwaite (1951) provides a review of several latent learning experiments back then.

Apart from that, cognitive psychology recognizes that learning can only take place if there is a need present that needs to be satisfied. Due to the observation of latent learning, consequently, a need for certain, correct anticipations must be present in higher animals. The theory of anticipatory behavioral control evolved out of this conclusion

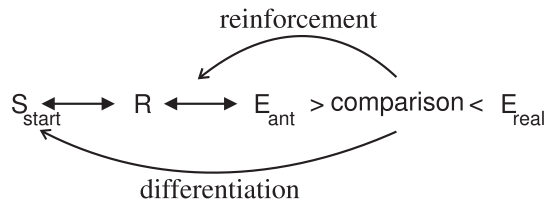

Figure 1. The anticipatory learning process in the ACS is derived from the visualized cognitive learning theory of anticipatory behavioral control.

(Hoffmann, 1993). The theory suspects that (1) any behavioral act or response (R) is accompanied with an anticipation of its consequences; (2) the anticipations of the effects E_{ant} are compared with the real effects E_{real}; (3) the relation between response and anticipation is strengthened when the anticipations were correct and weakened otherwise; (4) inaccurate anticipations lead to a further differentiation of the conditions. Figure 1 visualizes the basic idea of this theory. As is shown later on, the ACS implements this learning idea in the anticipatory learning process (ALP) in combination with the learning classifier system framework.

In relation to mathematics and physics, Rosen (1985) placed the idea of anticipations and their observations in a mathematical framework. Moreover, he stressed that anticipations are an essential part of distinguishing living systems from machines (Rosen, 1991). Despite the conflicting theory that the current behavior of any anticipatory system can be simulated by a pure reactive system, adaptation is often not possible without anticipations. Drescher (1991) stressed this insight as well and emphasizes that it is essentially easier to learn three part rules rather than two part rules in order to mimic condition - action - anticipation relations.

Related Artificial Systems

Drescher (1991) devised an anticipatory artificial learning system which is closely related to the ACS. Similar to the ACS, his rules, he called schemata, have three main parts: context, action, and result. Conditions and anticipations are represented by zero or more possibly negated items. Drescher relates his approach to Piaget's development theory of infants. He combines many processes observed in the different development stages in infants in his system, which makes it hard to clarify and analyze the actual interactions of the processes. Nonetheless, his experiment impressively shows parts of the development stages in an environmental setting that mimics the basics of infants perceptions and possible actions. While his generation of offspring rules is mainly based on statistics, rule generation in the ACS is based on the above described theory of anticipatory behavioral control realized in the ALP. Moreover, the 'chicken-and-egg' problem of whether to first specify consequences or conditions, approaches the ALP by its direct specialization of consequences and conditions when changes occur, assuming relevance in both parts.

In the reinforcement learning field, the Dyna architecture (Sutton, 1991b) reveals similarities with the ACS architecture. Although Sutton (1991a) keeps the introduction of Dyna explicitly independent from the model used, tabular approaches with completely specified environments were published. Similar to the ACS, Dyna forms its internal environmental model from scratch, initially assuming that actions don't cause any effects; however, usually explicit global knowledge about the environment is provided to Dyna, while the ACS only deals with local perceptions and a set of available actions. Moreover, Dyna approaches known to us that are dealing with generalization are generalizing over the coding which is specified before the start of the experiment as for example in a tile coding approach in Kuvayev and Sutton (1996). Even without a GA, the ACS forms a generalized environmental representation from scratch without any additional knowledge. Note that nonetheless, Dyna ideas can be applied to the ACS and indeed, a modified Dyna algorithm called the one-step mental acting algorithm was successfully applied in the ACS (Stolzmann et al., 2000).

The ACS combines the idea of learning by anticipations, realized in the ALP, with the LCS framework (Booker et al., 1989). In the learning classifier system community, Holland and Reitman (1978) outlined the possibility of evolving environmental representations with a classifier system. Later, Holland (1990) proposed an implicit model building approach. The idea was to use tags that specify if a current 'action' posted to a message list actually specifies an action or an anticipation. Riolo (1991) implemented this idea in his CFSC2 classifier system and was able to show latent learning capabilities. Moreover, he was able to show the possibility of lookahead-planning in his system and proposed multi-step predictions in order to refine the policy and consequently increase performance. However, the implicit formation due to the tags appeared to be misleading. Although implemented in a learning classifier system, Riolo's mechanism did not actually achieve significant generalization capabilities, and no further research has been reported on CFSC2.

Applying a more explicit linkage between classifiers in LCSs, Tomlinson and Bull (2001) show improvement in experiments with a corporate ZCS and XCS implementation (CZCS, CXCS). The approach links rules probabilistically resulting in the formation of cooperations among classifiers. The linkage consequently evolves an implicit representation of an effect or 'expecton' by the linked successive condition. This allows the use of anticipatory processes. A problem in the linkage formation seems to be the difference between linkage and reward space. Also, their evolution of linkage is policy dependent.

By contrast to the CFSC2 framework and CXCS, the ACS forms explicit condition-action-effect rules. The following section provides a precise description of the ACS. A more detailed discussion of the cognitive origins and capabilities of the ACS can be found in Stolzmann et al. (2000).

The Anticipatory Classifier System

As explained above, the ACS has its origins in LCSs and cognitive psychology. In the following sections we first explain the general framework of the ACS. Next, we explain the ALP and by using a simple environment illustrate the different parts of it. Finally, the limits of the ALP and especially the causes for over-specialization are disclosed by means of the introduced environment.

The Framework

The framework of the ACS shows many similarities with LCSs. Differences can be detected in the enhanced classifier structure as well as the application of the learning process. We start our explanations with the environmental interaction, proceed to the knowledge representation, and finally define one behavioral act in the system.

Environmental Interaction

Similar to all unsupervised learning approaches and especially LCSs, the ACS interacts autonomously with an environment. In a behavioral act at a certain time t it perceives a situation $\sigma(t) \in \mathcal{I} = \{\iota_1, \iota_2, ..., \iota_m\}^L$ where m is the number of possible values of each environmental attribute (or feature), $\iota_1, ..., \iota_m$ are the different possible values of each attribute and L is the string length. Note, that each attribute is not necessarily coded binary but can only take discrete values. Moreover, the system can act upon the environment with an action $\alpha(t) \in \mathcal{A} = \{\alpha_1, \alpha_2, ..., \alpha_n\}$ where n specifies the number of different possible actions in the environment and $\alpha_1, ..., \alpha_n$ are the different possible actions. After the execution of an action, the environment provides a scalar reward $\rho(t) \in \Re$.

Knowledge Representation
As in other LCSs the knowledge in the ACS is represented by a population $[P]$ of classifiers. Each classifier represents a condition-action-anticipation rule having the following components:

- The *condition part* (C) specifies the set of input states (situations) in which the classifier can be applied.
- The *action part* (A) proposes the classifier's action.
- The *effect part* (E) anticipates the effects that the classifier 'believes' to be caused by the specified action.
- The *mark* (M) records the values of each perceptual attribute of all situations where the classifier does not anticipate correctly.
- The *quality* (q) measures the accuracy of the anticipations.
- The *reward prediction* (r) predicts the reward expected after the execution of action A.

The condition and effect part consist of the values perceived from the environment and '#'-symbols (i.e. $C, E \in \{\iota_1, ..., \iota_m, \#\}^L$). A #-symbol in the condition called 'don't-care'-symbol means that the classifier matches any value in this attribute. However, a '#'-symbol in the effect part called 'pass-through'-symbol means that the classifier anticipates that the value of this attribute won't change after the execution of the specified action A. An action can be any action possible in the environment $(A \in \mathcal{A})$. The *mark* has the structure $M = (m_1, ..., m_L)$ with $m_i \subseteq \{\iota_1, ..., \iota_m\}$. The measures q and r are two scalar values where $q \in [0, 1]$ and $r \in \Re$. A classifier with a quality greater than the threshold θ_r is called *reliable* and becomes part of the internal environmental model. A classifier with a quality q lower than the threshold θ_i is considered as inadequate and is consequently deleted. All these parts are modified according to the reinforcement learning mechanism and the ALP explained later in this section.

A Behavioral Act
The ACS starts with a population of most general classifiers, i.e. there exists a classifier for each possible action that has only '#'-symbols in the condition and effect part. This means that the ACS 'believes' in the beginning that each action won't have any effect in the environment. In order to ensure the presence of a classifier in each presented situation, the initial classifiers have a special status in the population and are never deleted.

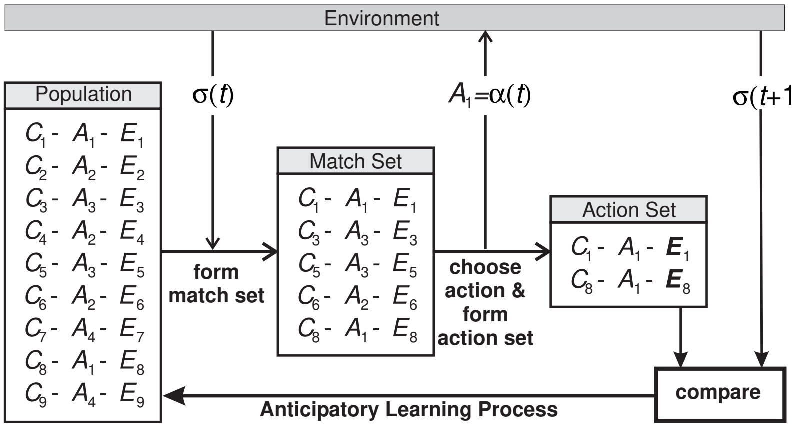

Figure 2. A behavioral act visualized with ALP application

A behavioral act, visualized in figure 2, starts with the perception of the current situation $\sigma(t)$. First, the ACS forms a match set $[M](t)$ out of $[P]$. $[M](t)$ consists of all classifiers in $[P]$ that match $\sigma(t)$ (i.e. all 'care'-symbols are identical to the corresponding position in $\sigma(t)$). Next, an action $\alpha(t)$ is chosen applying some behavioral policy in the match set [M]. Here, we usually apply an ϵ-greedy policy π (Sutton and Barto, 1998) with a high value for ϵ to increase the tendency of exploration. This policy can be written as,

$$\pi = \begin{cases} action(cl) : cl = \arg\max_{cl \in [M]} q_{cl} \cdot r_{cl} & \text{if random number} < \epsilon \\ \text{random } A \in \mathcal{A} & \text{otherwise} \end{cases} \tag{1}$$

where $\arg\max_{cl \in [M]}$ denotes the classifier $cl \in [M]$ with which the equation that follows is maximized. When exploiting the evolving knowledge (i.e. with a probability ϵ), the action of the classifier is chosen whose quality multiplied with the reward prediction is the highest. Otherwise, a random possible action is chosen. In the rare case of two identical maximal $q \cdot r$ values, the first classifier is chosen. With the chosen action $\alpha(t)$ an action set $[A](t)$ is then formed, consisting of all classifiers in $[M](t)$ that propose action $\alpha(t)$. After the execution of $\alpha(t)$, the perception of $\rho(t)$, the perception of the next situation $\sigma(t+1)$, and the formation of the resulting match set $[M](t+1)$, $[A](t)$ is modified. The modification covers the application of a reinforcement learning technique (considering $\rho(t)$ and $\max_{cl \in [M](t)} (q_{cl} \cdot r_{cl})$) as well as the application of the ALP (taking $\sigma(t+1)$ into account). Since all our experiments are divided into trials such behavioral acts are executed until one trial ends. Behavioral acts in one trial can be written in pseudo-code as follows:

EXECUTE ONE TRIAL ([P]):

```
t ← 0
PERCEIVE SITUATION σ(t)
DO{
    GENERATE MATCH SET [M](t) out of [P] using σ(t)
    if(t > 0)
        APPLY REINFORCEMENT LEARNING in [A](t−1) using
                  ρ(t−1) and max_{cl∈[M](t)} (q_cl · r_cl)
        APPLY ALP in [A](t−1) considering σ(t) modifying [P]
    CHOOSE ACTION act with the ε-greedy policy π
    GENERATE ACTION SET [A](t) out of [M](t) according to act
    EXECUTE ACTION act
    PERCEIVE REWARD ρ(t)
    PERCEIVE SITUATION σ(t+1)
    if(end of one trial)
        APPLY REINFORCEMENT LEARNING in [A](t) using ρ(t)
        APPLY ALP in [A](t) considering σ(t+1) modifying [P]
    t ← t+1
}while(not end of one trial)
```

Reinforcement Learning

As in XCS, our reinforcement learning approach adapts the Q-learning idea in reinforcement learning (Watkins and Dayan, 1992) to the ACS framework. This step away from the traditional bucket brigade (Holland, 1985) which is comparable with Sarsa in the reinforcement learning field (Sutton and Barto, 1998) enables a general policy independence of the reward learning in ACS and LCSs in general. A first mathematical analysis of Q-learning in generalizing systems such as LCSs can be found in Lanzi (2000b).

In order to learn an optimal policy in the ACS, the reward prediction r of each classifier in an action set is updated. For the reliability of the maximal Q-value in the successive state, we consider the quality of the classifier assuming that the reward converges in common with the accuracy of the anticipation. Once an accurate, reward sufficient model of the environment is evolved, the update method converges to the Q-learning theory and the formation of an optimal policy is assured.

$$r = r + b_r \cdot (\rho(t) + \gamma \cdot \max_{cl\in[M](t+1)} (q_{cl} \cdot r_{cl}) - r) \quad (2)$$

The parameter $b_r \in [0,1]$ denotes the reward learning rate and $\gamma \in [0,1]$ the discount factor similar to Q-learning.

The Anticipatory Learning Process

As mentioned before, the ALP is derived from the psychological learning theory of anticipatory behavioral control (Hoffmann, 1993). While the similarity with the learning theory is discussed elsewhere (Stolzmann et al., 2000), this section explains each inherited process in detail. Next, we introduce a simple environment for the ACS, in order to illustrate each process as well as to reveal its motivation. Moreover, we illustrate the causes for over-specialization with this environment.

The Process in Detail

The ALP compares the anticipation of each classifier in an action set with the real next situation $\sigma(t+1)$. According to this comparison and the current structure of the classifier, the classifier is modified and a new classifier may be generated. If a generated classifier already exists in the population, the new classifier is not inserted in the population but the quality of the old classifier is increased applying the Widrow-Hoff delta rule (Widrow and Hoff, 1960).

$$q_{cl} = q_{cl} + b_q \cdot (1 - q_{cl}) \tag{3}$$

The parameter $b_q \in [0,1]$ denotes the anticipatory learning rate. Note that the ALP consequently never creates identical classifiers in the population. The quality q of a generated classifier is set to the parent's value but never lower than 0.5 to give the new classifier a chance to establish itself in the population. The reward prediction r always inherits the value of its parent. In the following, we explain the modification and off-spring generation in further detail.

Useless Case Due to its origins in cognitive psychology where basically any action leads to some (at least proprioceptive) change in perceptions, cases where no change is perceived from the environment (i.e. $\sigma(t+1) = \sigma(t)$) are handled in the *useless case*. In this case the ALP decreases the quality of all classifiers in the action set using the Widrow-Hoff delta rule.

$$q_{cl} = q_{cl} - b_q \cdot q_{cl} \tag{4}$$

Moreover, each classifier is marked by the situation $\sigma(t) = \{\iota_1, ..., \iota_L\}$ it was applied in (i.e. $M' = (m'_1, m'_2, ..., m'_L)$ with $m'_i = m_i \cup \{\iota_i\}$). Note however, that the initial, completely general classifiers are never marked. Applying the useless case ensures that all classifiers in the ACS except for the initial, completely general classifiers predict changes in the environment and should only be applicable when the predicted change occurs. Thus, the evolving model only specifies condition action pairs that cause changes in the environment.

The comparison of anticipation and resulting state can be divided into two further cases which we explain below.

The Unexpected Case In the unexpected case, a classifier does not anticipate the resulting state correctly. This is the case when one or more of the predicted values are incorrect. In this case the classifier is marked by situation $\sigma(t)$ as formulized in the useless case. Moreover, its quality is decreased using equation 4.

After the modification of the classifier a new classifier may be generated if it is possible to form a correct anticipation out of the effect part E of the old, parental classifier by only changing pass-through-symbols to specific values. Hereby the ALP assumes that changing attributes are relevant in the conditions and consequently specifies those attributes in the conditions as well.

The Expected Case In the expected case, a classifier anticipates the results of an action correctly. Essentially, each attribute with a pass-through symbol in the effect part stays the same, and each attribute specified in the effect part changes or stays at the specified value. If in this case the classifier does not procreate offspring, its quality is increased applying again the Widrow-Hoff delta rule with anticipatory learning rate b_q as specified in equation 3.

Moreover, if the classifier is marked and differences can be found between the mark and the situation the classifier was applied in, a new classifier is generated that specifies a part or all of those differences in its conditions. Maximally u_{max} of such 'unchanging' attributes are allowed to be specified in the conditions in this way. Remember that the mark specifies the characteristics of the state(s) where the classifier did not anticipate correctly. Thus, a distinction from those states creates classifiers that are more and more likely to be only applicable in situations where their specified anticipation is correct.

In the next section we illustrate the specialization in the unexpected and expected case on a simple gripper problem and essentially reveal the offspring generation in the expected case in more detail.

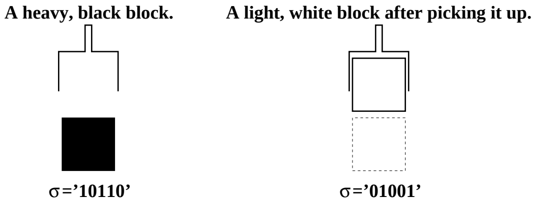

Figure 3. Using the illustrated gripper problem, we explain the intuition and function of each process in the ALP in more detail.

The ALP in Action: A Simple Gripper Problem

The problem shown in Figure 3 consists of a gripper that is able to lift a block that is situated under it. Since we are only interested in this one action and essentially a classifier that learns this action successfully this is the only possible action in the environment coded by an `1`. Thus, the environment is basically one-step. In each trial the environment presents a block to the gripper and the ACS tries to lift it. Our interest lies in the anticipations of the ACS—is the ACS able to correctly anticipate that it can only lift the block in certain conditions?

The coding is kept as simple as possible in order to show each mechanism. First of all, we code each situation in binary. The first bit specifies if the block is situated under the gripper. The second bit specifies if the gripper holds the block. The third bit denotes the color of the block and finally, the last two bits denote the weight of the block distinguishing between four weight levels.

As mentioned above, the ACS starts with a most general knowledge. Thus, in this environment in the beginning the population of the ACS consists of one classifier that predicts that nothing will change when executing the lifting action `1`, that is cl_1 =`#####-1-#####`. Let's assume that the ACS lifts the block successfully in the first trial resulting in an unexpected case since the block position changes. Thus, the quality of cl_1 is decreased and a new classifier is generated that specifies the first two bits in condition and action since both of them are changing, that is cl_2 =`10###-1-01###`.

Let's now assume that a lift operation is only successful if the weight of the present block is less than half of the provided measurement (i.e. the last two bits are `01` or `00`). Moreover, let's assume that the environment presented only white blocks so far (coded by `0`). Thus, cl_2 eventually has a mark of the form $M_{cl_2} = (1, 0, 0, 1, \{0, 1\})$. Note that we hereby and in the remainder don't print the braces of an one-element set to allow better readability. Back in an expected case (for example `10001`) a new classifier is generated that is specialized in the fourth component, cl_3 =`10#0#-1-01###`. This example reveals the idea of the mark, the two last bits appear to be relevant for the successful execution of the action, however, the definite difference in the fourth bit is stronger than the occasional difference in the fifth bit and the forth bit is consequently specified.

Let's now assume that the environment provides always light, white blocks or heavy, black ones. In this case the mark of cl_2 eventually looks like this: $M_{cl_2} = (1, 0, 1, 1, \{0, 1\})$. Back in an expected case (for example `10001`) it is not clear if either the third bit (the color bit) or the fourth bit (the definitely relevant weight bit) should be specialized and—since both appear to be sufficient for a successful application—the ALP specifies either of the two (that is cl_3 =`100##-l-01###` or cl_3 =`10#1#-l-01###`).

The last marking case that leads to a specialization is illustrated as follows. Let's assume that a lifting action is only successful when the block has the lightest weight. Moreover, the environment provides black or white blocks randomly where the color has no relation to the weight of the block. In this case the mark of cl_2 eventually looks like this: $M_{cl_2} = (1, 0, \{0, 1\}, \{0, 1\}, \{0, 1\})$. Back in an expected case (for example `10100`) it is not clear which bit is/are best to specialize but apparently none of the bits make sense to specialize on their own. Thus, the ALP specializes all of the bits at once resulting in cl_3 =`10100-l-01###`. This is certainly a crucial case and a more gradual method of specializing only two bits in this case (since the specification of at least two further bits is definitely necessary) is imaginable. However, experiments with such a gradual increase in specificity showed a similar performance but a slight increase in the population size due to the increased generation of not sufficiently specified classifiers. Moreover, in this case the specificity threshold u_{max} needs to be considered. If u_{max} would be less than three, e.g. two, then one of the three possible classifiers would be generated at random (that is cl_3 =`1010#-l-01###` or cl_3 =`101#0-l-01###` or cl_3 =`10#00-l-01###`).

Causes for Over-Specialization

The ALP starts with most general rules and gradually generates more specialized classifiers as explained above. By doing that a complete internal environmental representation evolves as shown elsewhere (Stolzmann and Butz, 2000). However, there are cases where the ALP makes assumptions which are not necessarily correct. These assumptions can lead to an over-specialization of classifiers in the conditions, i.e. attributes in the conditions get specialized which are irrelevant for a successful anticipation. Using the gripper example we reveal the cases where over-specialization occurs.

First, the ALP assumes that changing attributes are relevant in the conditions. In the gripper example the ALP generated the classifier

cl_2 that specified the first two bits in the conditions. This makes sense since it should be assured that a block is present and under the gripper. Indeed, considering our own environment, it appears to be reasonable to always assure that something is present before we can execute an action upon it (e.g. in order to be able to drink out of a glass, the glass must be present). However, in the gripper environment it is actually not necessary to specify those two bits since by definition of the environment there is always a block present. Moreover, actions that don't manipulate another entity but rather ourselves such as movements, fall in a different category and the specification of such changes in the conditions are consequently not always necessary.

Second, the specialization due to the mark can cause further over-specialization. (1) The so far experienced incorrect states can give incomplete information in the mark. (2) The limited structure of the mark can be misleading. (3) Non-determinism in the environment can cause unnecessary specialization. In the following we illustrate each of these three points.

The problem of incomplete information in the mark was illustrated in the gripper example above in which a wrong correlation between color and weight was detected. Another example in the gripper problem would e.g. be the case when only one incorrect state was experienced (e.g. $M_{cl_s} = (1,0,0,1,0)$) in the last setting of the gripper example above in which only the lightest block is liftable. Now let's assume that the next presented situation is `10100` which leads the ALP to a specialization of either the irrelevant color bit or the partially relevant second weight bit. When the color bit is specified, the classifier will cause a further over-specialized classifier, since the two weight bits still need to be specified and there is no generalization mechanism.

The limited structure of the mark is illustrated in the gripper example in the last setting where only the lightest block is liftable and the color bit is changing randomly. If the mark would store all unsuccessful states separately it would eventually look like this: $M_{cl_2} = \{(1,0,0,0,1),(1,0,1,0,1),(1,0,0,1,0),(1,0,1,1,0),(1,0,0,1,1),(1,0,1,1,1)\}$. Thus, a correct detection of the two relevant weight bits would be possible. However, since some over-general classifiers will eventually store nearly all possible environmental situations in their mark this is not an option.

Non-determinism in the environment is a broad problem and can severely influence the specialization. In a setting with stochastic actions the mark will essentially store the state in which the classifier is usually correct resulting in no useful information for specialization. In non-Markov problems the mark would again store the state in which the classifier is successful sometimes and the same problem occurs.

Randomly changing, irrelevant attributes in the perceptions can cause further confusion in the mark. Again, the mark would store states in that the classifier usually anticipates correctly. (An approach that detects the property in the three above cases is presented in Butz et al. (2001)). Moreover, in the case of perceptual mistakes the information in the mark becomes more and more useless. Finally, non-determinism could be caused by other animats or forces in the environment which appears to be the hardest challenge for the ACS in the current setting.

The causes show that it is necessary to introduce some kind of generalization mechanism. However, it becomes increasingly difficult to refine the ALP. Moreover, the diversity of the causes shows that it is extremely difficult—and in some of the cases probably impossible—to detect all of them in a deterministic way. Thus, with an evolutionary system in hand, we decided to apply a genetic algorithm (GA) to the ACS whose aim is to generalize over-specialized classifiers and consequently lead to the generation of accurate, maximally general classifiers, i.e. classifiers whose anticipations are always accurate while the conditions are as general as possible. The modified GA and its interaction with the ALP is introduced below.

Parameters in the ACS

Parameters in the ACS were introduced in the previous sections. Here, we summarize the parameters and their usage:

- The *inadequacy threshold* θ_i specifies when a classifier becomes inadequate and is deleted from the population. It is usually set to 0.1.

- The *reliability threshold* θ_r specifies when a classifier becomes reliable and consequently becomes part of the internal environmental model.It is usually set to 0.9.

- The *reinforcement learning rate* b_r is used in the reward prediction update of a classifier (equation 2). A learning rate of 0.05 is usually used.

- The *discount factor* γ discounts the maximal reward expected in the subsequent situation. It is usually set to 0.95.

- The *anticipatory learning rate* b_q is used in the quality update in equations 3 and 4. Usually, we use a rate of 0.05.

- The *specificity threshold* u_{max} specifies the maximally allowed number of specified attributes in the condition of a classifier that are anticipated to stay the same in the effect part.

– The *exploration probability* ϵ specifies the probability of choosing a random action during exploration according to the ϵ-greedy policy in reinforcement learning.

Genetic Generalization in the ACS

The last section showed that the ALP sometimes specifies irrelevant attributes in the conditions of a classifier. Since there are several causes and several types of over-specialization and since the ALP mechanism generates only specialized offspring, the ALP can be viewed as a specialization pressure in the ACS. This section consequently introduces a genetic generalization pressure designed to fix the occurring ALP over-specialization. Parts of this work were published previously in Butz et al. (2000a) and Butz et al. (2000c). However, the explanations herein are further detailed, the mechanisms are refined and the results are enhanced.

First, we provide the idea of how to realize the generalization pressure using a genetic algorithm. Next, a precise description of the algorithm is provided. Finally, we reveal interactions between ALP and GA in the ACS and introduce subsumption in order to assure a further convergence to the accurate, maximally general rules.

The GA Idea

The realization of the GA in the ACS was inspired by Wilson's XCS classifier system (Wilson, 1995). While we outline the GA method and the relation to XCS in this section, the next section describes each process in the GA in detail.

First, we have to consider where the GA should be applied. Since our GA is searching in the generality space of one specific set of situations - action - result triple we need to use a niche GA. Such a niche GA is applied in XCS as well. Since Wilson (1998), the GA in XCS takes place in the action set which showed an increase in performance and a further convergence to the (in XCS reward prediction) accurate, maximally general classifiers. Consequently, the GA in the ACS takes place in the *action set* as well, which actually represents the so far learned particular set of situations - action - result space. In order to control the frequency of the GA application in such an action set we introduce a *time stamp* to each classifier that records the last GA application in its set similar to XCS. Applying a GA in a set consequently depends on the average last GA application in the particular set determined by the time stamps of the classifiers in the set.

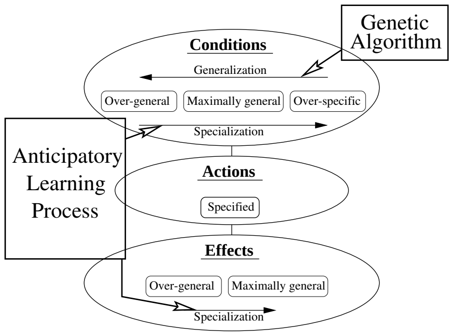

Figure 4. While the ALP specializes conditions and effects, the GA generalizes only conditions.

The next step is to get the genetic pressure right. As mentioned before, similar to the accuracy based fitness in XCS the ACS uses an accuracy measure of its (state) anticipations. This accuracy measure (represented by the quality q) can be used as the *fitness* measure in the selection of parents and the deletion of low accurate classifiers realizing the principle of the survival of the fittest and the die out of the weak. However, our GA is supposed to add a generalization pressure to the system rather than a more general exploration pressure. Thus, another criterion for the weakness of a classifier is its over-specialized condition part indicated by its *formal specificity* in comparison to other classifiers.

The genetic operators need to be modified as well. First of all, the question arises if the GA should be applied in condition, action and effect part or only in a part of it. Since our GA is intended to generalize conditions, the GA is only applied in the *conditions*. An enhanced application which alters actions and/or anticipations is imaginable. However, in this case the whole current GA idea would be thrown over. Our GA is intended to generalize conditions and not alter anticipations. In order to evolve new anticipations with a GA the pressures certainly need to be altered. Due to the intended generalization our mutation operator only *generalizes* conditions, i.e. change specified attributes back to 'don't care'-symbols. Crossover, in the fate of combining ideas and consequently realizing some sort of innovation (Goldberg, 2002), combines generalization ideas in the conditions using *two-point crossover*. To only combine such ideas of generalization classifiers are only crossed if they anticipate the same outcome. Figure 4 visualizes where the genetic generalization pressure acts in the ACS as well as reveals the difference from the ALP application.

Moreover, we need to find a criterion when to delete classifiers. XCS uses a fixed population size and deletes classifiers with a probability dependent on an action set size estimate of each classifier ensuring an equal distribution of classifiers in each environmental niche. This method is not similarly applicable in the ACS. Since the ALP generates in different niches different numbers of classifiers an action set size estimate in the ACS has no real GA relation and is consequently misleading. Thus, we decided to use a *fixed action set size* rather than a fixed population size and delete classifiers in the particular set once this measure is exceeded.

Finally, our GA is supposed to cause a convergence to the accurate, maximally general classifiers. This is realized in XCS by the fitness based on accuracy combined with the niche GA. Essentially, the accuracy measure prevents over-generalization while the niche GA favors accurate, more general classifiers for reproduction, since they occur more often in an action set as stated in Wilson's generalization hypothesis (Wilson, 1995) and later further investigated and enhanced to an optimality hypothesis (Kovacs, 1996, Kovacs, 1997). Moreover, the convergence was further optimized by the introduction of subsumption and the GA application in action sets (Wilson, 1998). Both pressures seem to be present in the GA of the ACS as well, however additional to the GA reproduction pressure there is the reproduction of classifiers due to the ALP. Thus, we altered the deletion pressure as mentioned above in order to favor formal generality.

Finally, we need to allow identical classifiers in the population since each action set should eventually be populated by only accurate, maximally general classifiers. In order to accomplish an efficient representation of identical classifiers we enable a classifier to actually represent more than one, identical classifiers. Moreover, we use subsumption similar to (Wilson, 1998) in the GA and the ALP in order to stress the convergence further.

How the GA Works

All the above mentioned processes are formally described now. The next section then explains how the GA interacts with the ALP.

Since the GA relies on the quality measure q the ALP takes place before the GA application. This assures that all classifier parameters are set to the most accurate value possible. The GA mechanism works as follows:

- First, it is determined if a GA should take place in the current action set. For that, each classifier records the last time it was in an action set where a GA took place in the GA time-stamp t_{ga}. A GA is applied, if the actual time t minus the average of all time-stamps t_{ga} is bigger than the threshold θ_{ga}. The higher this threshold, the more the ALP rules the evolutionary process.

- If a GA is applied, two classifiers are selected by roulette wheel selection. Similar to Wilson (2000) we emphasize the selection of highly accurate classifiers by applying a power function. Thus, we determine the fitness of each classifier by the cube of its quality (i.e. q^3).

- The parents stay in the population and compete with their offspring.

- With a probability μ, each specialized attribute in the conditions of each offspring is mutated to a 'don't-care'-symbol (i.e. generalized).

- If the two offspring-classifiers have identical effect parts they are crossed applying two-point crossover with a probability χ. In this case the reward prediction r and the quality q is averaged over the offspring.

- The inherited quality q of an offspring is multiplied by 0.5 in order to prevent over-generalization errors. The inherited strength stays the same.

- An offspring is only inserted in the population if the condition is not completely general.

- Each classifier keeps an additional numerosity num that counts how many identical 'micro'-classifiers this technically called macro-classifier represents. This is identical to the macro-classifier notation in XCS (Wilson, 1995). If a created classifier already exists in the population, the numerosity of the old classifier is increased by one and the new one is discarded.

- If the action set size exceeds the action set size threshold θ_{as} excess classifiers are deleted from the set.

- The deletion method deletes classifiers using tournament selection with a tournament size of 1/3 the action set size. The tournament is held in two stages. First, the quality is considered. If the quality differs by more than 0.1 the worse classifier is deleted. Otherwise, the more specialized classifier is deleted. For an helpful tournament selection analysis the interested reader should refer to Goldberg and Deb (1991).

The quality initialization of the offspring differs from previous publications (Butz et al., 2000a, Butz et al., 2000c). While the quality was previously rigorously set to 0.5 we adjust herein the fitness inheritance idea to our GA framework. The quality initialization also differs from the initialization in the ALP since the ALP directly specializes implying a classifier with a higher quality while the GA indirectly generalizes which can probably cause a loss in quality.

Crossover was modified as well. In Butz et al. (2000a) crossover was unrestricted and only used in the Woods2 test suite. However, the idea of crossover is to combine generalization ideas in each environmental niche and not to combine ideas of different niches. While the application of the GA in the action set already results in implicit niching, this is a further niche restriction which is similar to the idea of sharing (Goldberg and Richardson, 1987). In the Woods14 environment, investigated below, we analyze crossover and show the better performance of the restricted version.

Interaction of ALP and GA

Right now, the GA generalizes while the ALP proceeds with over-specialization. However, once an accurate, maximally general classifier is evolved, it won't be further modified but could possibly be deleted by the GA. In order to decrease the probability of losing the best classifier and ensure stable convergence to it in each environmental niche, we consequently allow equal classifiers (represented in one actual macro-classifier by the numerosity num). Thus, the GA together with the ALP evolve more and more of those best classifiers which eventually take over each environmental niche.

Despite this, the GA proceeds to over-generalize such accurate, maximally general classifiers and the ALP again over-specializes sometimes over-general classifiers. If the evolution of those disfavored classifiers is stronger than the tendency to produce the ideal classifier it still can happen that some of the ideal classifiers get lost. Therefore we now introduce subsumption which is designed to solve this error.

Subsumption deletion was previously introduced by Wilson (1998) in XCS. The idea is that once an accurate, maximally general classifier is found this classifier absorbs all other classifiers which are more specific than itself. The same subsumption mechanism is now applied in the GA and the ALP. A classifier cl_{sub} subsumes a new classifier cl_{new} if it is (1) reliable, (2) experienced, (3) not marked, (4) more general, (5) has the same action, and (6) anticipates the same. Classifier cl_{sub} is considered as reliable if its quality $q_{cl_{sub}}$ is greater than the reliability threshold θ_r. For the experienced criterion each classifier keeps an additional experience counter exp that records the number of ALP applications. Considering this counter, cl_{sub} is experienced if $exp_{cl_{sub}}$ is greater that the experience threshold θ_{exp}. The unmarked criterion further assures

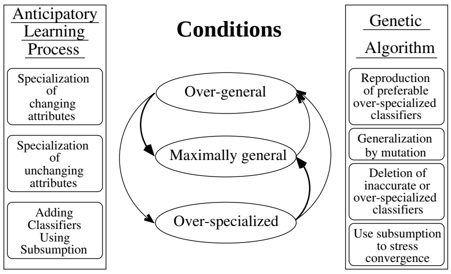

Figure 5. The different arrows visualize how the specialization due to the ALP and the generalization due to the GA act upon the conditions.

that cl_{sub} always anticipated correctly so far. For the generality criterion we consider the syntactic generality in the conditions. Thus, cl_{sub} is more general than cl_{new} if the condition part $C_{cl_{new}}$ has less '#'-symbols than $C_{cl_{sub}}$.

When cl_{new} is subsumed it is not inserted. Moreover, if the subsumption occurs in the ALP, the quality $q_{cl_{sub}}$ is increased using equation 3. If the subsumption occurs in the GA, the numerosity $num_{cl_{sub}}$ is increased by one. Thus, only the GA creates identical classifiers by increasing numerosities while the ALP stays consistent with its principle of not creating identical classifiers. This method ensures that the ALP does not generate over-specialized classifiers anymore, once the accurate, maximally general one is found. Moreover, the method results in a decrease of the specialization pressure later in the run. This results in an increase of the genetic pressure and consequently an increase in the reproduction pressure of accurate, more general classifiers. Figure 5 visualizes the interaction of the GA and the ALP in the conditions of the classifiers.

Additional GA and Subsumption Parameters

The above description of the GA and the subsumption mechanism has mentioned additional parameters in each classifier as well as in the ACS itself. We summarize the parameters below.

Additional classifier parameters are the following:

- The *GA time stamp* t_{ga} records the last time the GA took place in an action set to which this classifier belonged.

- The *experience counter* exp counts the number of times the classifier underwent the ALP.

- The *numerosity* num specifies the number of micro-classifiers this macro-classifier represents. This measure however does not affect the performance and is rather meant to increase the efficiency of the system.

Additional ACS parameters are:

- The *GA application threshold* θ_{ga} controls the frequency of a GA application in a set.
- The *mutation rate* μ specifies the probability of changing a specified attribute in the conditions to a don't care symbol. Since this is a directed generalization mutation, the mutation rate is higher than in the common GA applications.
- The *crossover probability* χ specifies the probability of applying crossover in the conditions of a classifier.
- The *action set size threshold* θ_{as} specifies the maximal number of classifiers in an action set.
- The *experienced threshold* θ_{exp} specifies the least experience of a classifier in order to be a subsumer.

Typical parameter settings are given in the next section.

Performance of the ACS with a GA

Although the genetic generalization approach in the last section is based on the existing population and different sets in the ACS revealing many similarities with XCS, it is certainly not straight forward to see that the GA is any good. We are interested to know whether the GA is able to overcome the observed over-specialization. Does the GA result in an evolution of and a convergence to the aspired accurate, maximally general classifiers in the population or is it too strong, which would result in an over-generalized population, or too weak, which would result in a larger, over-specialized population?

This section empirically confirms that the GA is able to work well with the ALP and cause the population to converge to the target classifiers. In order to do that, we first present the resulting classifier list without and with GA application in our gripper problem. Next, we investigate the ACS performance in an application to the multiplexer test function. Finally, we provide model and policy learning results in maze environments.

Unless stated differently the parameters were set to the following values: The inadequacy threshold $\theta_i = 0.1$ and the reliability threshold

$\theta_r = 0.9$ are the usual values. The learning rates are set quite low to $b_r = b_q = 0.05$ in order to prevent errors. The discount factor is set quite high in order to allow long chaining: $\gamma = 0.95$. The number of specified, unchanging attributes is usually unrestricted, $u_{max} = \infty$. Encouraging fast model learning, we set the policy to pure exploration $\epsilon = 1.0$. The GA parameters are: $\theta_{ga} = 25$ in the gripper and multiplexer problem and $\theta_{ga} = 100$ in the maze problem. Although the threshold was set higher in the maze problems this does not cause any major impact on performance as shown in figure 13. Mutation is chosen unusually high since we deal with a directed generalizing mutation $\mu = 0.3$. Crossover is set to the standard value of $\chi = 0.8$. The action set size threshold showed to be robust with $\theta_{as} = 20$. A higher value would result in a higher growth in the population early in the run but increase the reliability of convergence later in the run. The experience threshold minimizes the error in subsumption and was chosen accordingly to the XCS value $\theta_{exp} = 20$. All results presented herein are averaged over twenty runs. The horizontal distances between the points in the graphs reveal over how many last steps or trials the performance is averaged. Population size curves are always plotted in terms of the macro-classifiers' size.

Accurate, Maximally General Classifiers in the Gripper Problem

Despite the relative simplicity of the gripper problem (introduced above) we identified many possible causes for over-specialization. First, we realized that the specialization in the condition of changing attributes is not always necessary in the environment. Next, we observed the limits in the mark which possibly resulted in an unnecessary specialization of the color attribute.

Indeed, these causes can be observed in a resulting classifier list, applying the ACS to the above described problem where we added a second action `r` for releasing the block in addition to the lifting action `l`. Figure 6 shows that classifiers evolved that are over-specialized in the color attribute. Shown is the setting where only the lightest block was liftable (weight attribute 00) and where black and white blocks were presented independently from the weight of the blocks. Monitoring figure 6 in more detail we see that the first classifier is marked and has low quality, revealing its possible incorrectness. Moreover, the mark does not identify the relevant weight bits. Classifiers two and three are definitely over-specialized due to the irrelevantly specified color bit. Classifier four is accurate, but could be more general in this environmental setting by ignoring one of the first two bits. Releasing a block is always successful in our environment once the block is lifted and consequently easily specified by classifier five. However, also this classifier could be more general by ignoring bit one or two. Classifiers six and seven are the initial classifiers that always remain in the population.

no	condition	action	effect	quality	mark	num
1	10###	l	01###	0.2	$(1, 0, \{0, 1\}, \{0, 1\}, \{0, 1\})$	1
2	10100	l	01###	1.0	-	1
3	10000	l	01###	1.0	-	1
4	10#00	l	01###	1.0	-	1
5	01###	r	10###	1.0	-	1
6	#####	l	#####	0.1	-	1
7	#####	r	#####	0.1	-	1

Figure 6. The resulting classifier list of an ACS without GA application in the gripper problem shows over-specialized classifiers.

The classifiers in the table show the predicted over-specialization of irrelevant changing attributes and the color attribute. Note also that the numerosity is always one since the ALP does not generate identical classifiers. Running this same setting with the GA should evolve a population with two accurate, maximally general classifiers that occupy the main part of the population, one for gripping and one for releasing.

Figure 7 shows this prediction in the resulting population when the ACS with GA is applied to the problem. Classifiers seven and eight have a much higher numerosity than the other classifiers and consequently overrule them. Note, that both classifiers are accurate, and maximally general. Classifier eight makes sure that releasing causes a change by specifying the presence of a block in its hand. Classifier seven only specifies the presence of a block on the ground and the lightest weight. Although classifier six is as accurate and maximally general as classifier seven, in this case classifier seven rules the population due to its much higher numerosity. In other runs the numerosity values were actually switched what reveals the continuous competition between the two classifiers in the GA. Classifiers one to five are all over-general indicating the continuous search and interaction of GA and ALP in the generality space.

The Multiplexer Problem

Although the last section demonstrated the capability of the GA to cause the ACS to generate accurate, maximally general classifiers, a decrease in the population size was not observed. Moreover, the gripper

no	condition	action	effect	quality	mark	num
1	10###	l	01###	0.4	$1, 0, \{0,1\}, \{0,1\}, \{0,1\}$	3
2	###00	l	01###	0.5	$0, 1, 1, 0, 0$	1
3	1#0#0	l	01###	0.4	$1, 0, 0, 1, 0$	2
4	1###0	l	01###	0.3	$1, 0, \{0,1\}, 1, 0$	1
5	##000	l	01###	0.4	$0, 1, 0, 0, 0$	1
6	#0#00	l	01###	1.0	-	1
7	1##00	l	01###	1.0	-	13
8	#1###	r	10###	1.0	-	19
9	#####	l	#####	0.1	-	1
10	#####	r	#####	0.1	-	1

Figure 7. When the GA is applied in the ACS, the accurate, maximally general classifiers evolve and over-rule others.

problem is certainly a simplified toy problem and only applicable for simple analysis.

Thus, in order to study the generalization capabilities of the ACS further, we decided to test it in an hard classification task which has been well studied before with XCS (Wilson, 1995,Wilson, 1998). Although the ACS is meant rather for the learning of a multi-step model, the multiplexer problem serves well to investigate the generalization capabilities of the ACS. Moreover, we show that the performance of the ACS is comparable with the XCS performance in this environment.

The multiplexer problem is essentially described by a multiplexer function whose complexity increases exponentially with the number of relevant attributes. The function is defined for lengths $l = k + 2^k$ ($k \in \mathbb{N}$) where the first k bits address one bit in the 2^k remaining bits. The value of the addressed bit is the return value of the function. For example, considering the 6-multiplexer with its 2 address bits, the solution of problem instance '011011' is 0 since the first two bits address the '0' bit in the remaining four. Each multiplexer function can also be written in disjunctive normal form with an in k exponentially increasing number of terms. For the 6-multiplexer the DNF representation is as follows:

$$6MP(x_1, x_2, x_3, x_4, x_5, x_6) = \neg x_1 \neg x_2 x_3 \vee \neg x_1 x_2 x_4 \vee x_1 \neg x_2 x_5 \vee x_1 x_2 x_6 \tag{5}$$

Since the ACS learns from its anticipations, we need to introduce a *perceptual causality* into the environment to make it solvable for the ACS. Consequently, we augment the string with additional information about the correctness of a classification (i.e. actions $\mathcal{A} = \{0, 1\}$). Either, we add only one attribute (thus, $L = l + 1$) which is 0 in each problem instance and switches to 2 if the correct action was executed and to 1 otherwise (i.e. $\mathcal{I}_{MP1} = \{0,1\}^{k+2^k} \cdot \{0,1,2\}^1$) or we add $2+k$ bits, distinguishing each problem instance in correctness as well as in the referred position as described in Wilson (1995) (i.e. $\mathcal{I}_{MP2} = \{0,1\}^{2+2k+2^k}$). The formula for this perceptual 'reward map' is defined by 1+ (value of the k position bits) + (return value) $+11\cdot$ correctness. For example, in the six multiplexer the problem instance '0100000000' with classification '0' would result in '0100000101' while a classification of '1' would result in '0100000010'. If the correct action was executed, the ACS receives a reward $\rho = 1000$. After that, one trial ends and the environment generates another random string.

We test the ACS in the multiplexer problem with both codings. In the next section we investigate whether the ACS is able to anticipate the result of its action correctly, that is, whether it evolves a correct internal model of the problem. Next, we investigate the reward learning capabilities in this setting providing a comparison to the performance of XCS.

Evolution of a Multiplexer Model

To monitor the evolution of a model in the multiplexer problem, we create, after each exploration trial, a random problem instance and compare the anticipation of the highest quality classifier with the real result for both actions. In all multiplexer settings we restricted the u_{max} parameter to prevent rigorous over-specialization.

Figure 8 compares the model formation in the setting with and without GA. Although both graphs reveal that the ACS is able to learn the model with and without GA, the difference in population size is very high. While the ACS without GA eventually learns each specific example due to over-specializing errors of the ALP, the GA is able to fix the over-specialization and causes the population size to converge to a low level. In the setting with only one additional attribute, also the knowledge increases much faster.

In the left of figure 9, it can be observed that performance of the ACS with GA relies only slightly on the choice of u_{max}. The parameter u_{max} restricts the specialization pressure of the ALP (that is, it restricts the number of specified attributes in the conditions which are anticipated not to change). Choosing u_{max} too low results in a strong genetic generalization pressure that gives the ALP no time to fix over-general classifiers. On the other hand, when choosing u_{max} too high, a loss in speed due to more function evaluations and the increase in population size can be observed. However, in this case the ACS was still able to reach 100% performance.

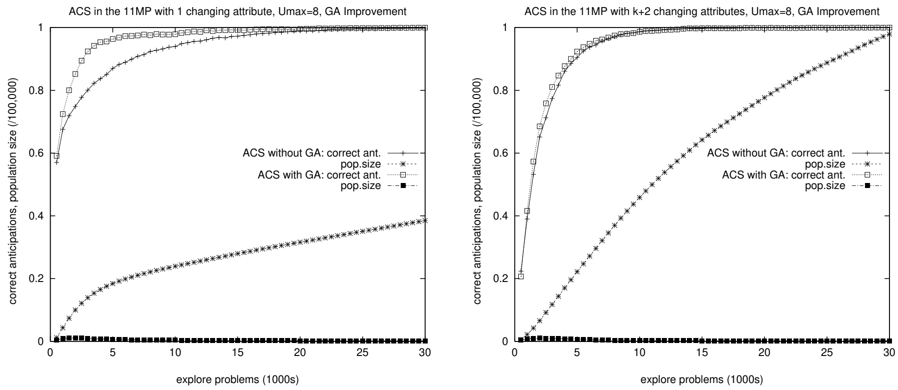

Figure 8. Comparison of ACS without and with GA in the 11-Multiplexer with the two perceptual codings.

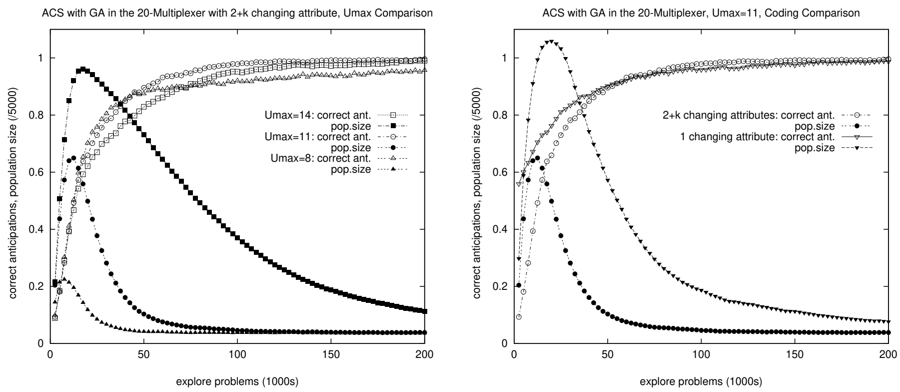

Figure 9. Comparison of different u_{max} values and the two perceptual consequence codings in the 20-Multiplexer.

Finally, we investigate the reliance of ACS on the type of coding. Since the ACS is essentially based on learning from environmental changes, the more information is extractable from changes in the environment, the easier it is for the ACS to learn an environmental representation. This fact can be observed in the left-hand side of figure 9. The ACS performs better when $2 + k$ additional attributes are provided since the number of problems to reach a 100% performance is smaller and the population size stays smaller. Due to the stronger separation of the problem space, the ACS is able to distinguish the different levels faster and consequently performs better.

Classification Learning
While the graphs in the last section demonstrated the evolution of a correct environmental model, this section investigates the classification

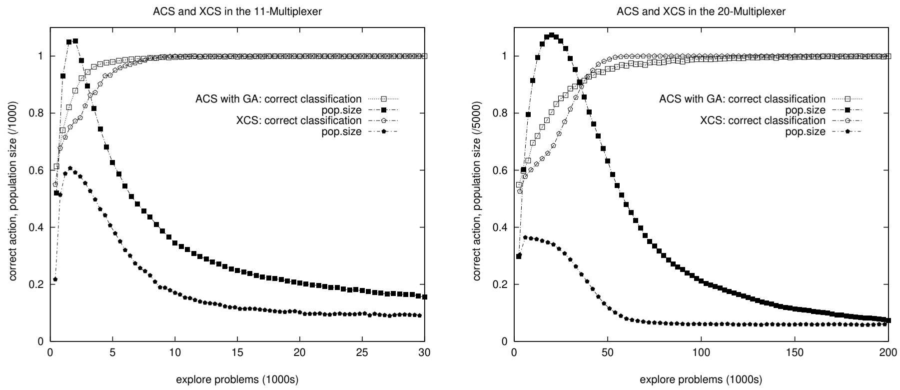

Figure 10. Comparison of ACS and XCS in the 11- and 20-multiplexer.

capabilities of the ACS. Although the ACS is meant to learn optimal behavioral policies in multi-step problems while the environmental model evolves, the ACS can be applied in the multiplexer setting as a classifier. The multiplexer environment now always provides a reward of $\rho = 1000$ if the classification is correct. The performance is monitored as in XCS (Wilson, 1998) by altering between one exploration and one exploitation step. During exploration the normal behavioral learning act is executed. However, during exploitation always the best action is chosen as classification, no learning takes place, and the performance is monitored according to the correctness of the classification.

Figure 10 reveals the performance of ACS in the 11 and 20 multiplexer problem comparing it to the performance of XCS. This comparison allows two important observations: (1) The ACS is able to classify the problems correctly after a number of instances similar to XCS; however, ACS certainly applies much unnecessary effort to accomplish this comparable result. (2) The population of ACS converges to a size similar to XCS. This convergence was studied for XCS in detail before (Kovacs, 1997, Wilson, 1998) and revealed that XCS converges to the accurate, maximally general classifiers. Thus, this shows that the GA in ACS is able to cause convergence to the accurate, maximally general classifiers. A more detailed examination indeed revealed the presence of the accurate, maximally general classifiers.

Maze Problems

Although the multiplexer problem in the last section is a great challenge and suitable for performance analysis and analysis of ACS's generalization capabilities, the ACS is designed as a multi-step system. It is intended to learn an environmental representation of an environment and to adapt its behavior with this model. Thus, we investigate now a more typical multi-step problem.

Maze problems, as we study them herein, have the framework of a discrete, two-dimensional environment. Each field can be occupied by either an obstacle or food or it can be empty. Perceptions are coded as a string of eight attributes coding the properties of each neighboring field starting in the north and coding clockwise. An empty field is coded by a '.', an obstacle with an 'O', and food with an 'F' which results in a perceptual space of $\mathcal{I}_{maze} \subseteq \{., O, F\}^L$. Thus, an animat located at the food position in Maze6 would e.g. experience the perception 'OOOO.OOO'. Primitive actions for the movement to each of the eight neighboring fields were provided resulting in an action space $\mathcal{A}_{maze} = \{N, NE, E, SE, S, SW, W, NW\}$. If an action leads to a field which is blocked by an obstacle the action has no effect. Food is provided at one position in the maze. Once the food is reached, a reward $\rho = 1000$ is provided, the *trial* ends, and the system is reset to a random position in the maze. Figure 11 shows the two mazes tested: Maze6 on the left-hand side and Woods14 on the right-hand side.

Our results are divided into two sections. First, we present model learning experiments, where we plot the evolving model and the population size. Second, we present reinforcement learning experiments where we plot the number of steps used in each trial and also the population size.

Evolution of a Model

As discussed above, a representation of anticipation automatically leads to the formation of an internal model of an environment. Consequently, the ACS also implicitly evolves a model of its perceived environment. Due to the anticipations in each classifier the classifiers form an implicit network of relations. The evolution, accuracy, and completeness of this model is studied herein. First, we explain how the evolving model is measured. Next, we present results in Maze6 and Woods14 comparing the ACS with and without GA as well as revealing the sensitivity of some GA parameters.

Performance Measure A special test is used to investigate the evolved model representation. Since we regard only reliable classifiers (i.e. classifiers with a quality q of more than θ_r) as part of the internal model, only those are examined in the performance test. For each possible environmental situation-action-effect triple in which a change occurs, the list of reliable classifiers is scanned for one that matches the situation, specifies the action and predicts the effect correctly. If such a classifier

O	O	O	O	O	O	O	O	O
O						O	F	O
O			O		O	O		O
O		O						O
O				O	O			O
O		O		O			O	O
O		O						O
O						O		O
O	O	O	O	O	O	O	O	O

O	O	O	O	O	O	O	O	O	O	O	O	O
O	O				O	O	O	O		O	O	
O		O	O	O		O	O		O		O	
O		O	O	O		O		O	O	O		O
O	F	O	O	O		O	O		O	O	O	O
O	O	O	O	O	O			O	O	O	O	O

Figure 11. The two tested mazes: Maze6 and Woods14.

exists, the triple is regarded as correctly represented. The performance measure reflects the percentage of such correctly represented triples.

Model in Maze6 Maze6 was previously investigated in Lanzi (1999) with XCS. He showed that the reward learning abilities of XCS can suffer due to the unequal reward-distribution and the unequal frequencies of encountering each state in the maze. While XCS with some modifications was able to form a complete and generalized reward-prediction model of the environment, in this section we are rather interested in the evolving anticipatory model. Furthermore we want to reveal the generalization capabilities of the GA in ACS. Thus we added in some cases additional bits $ABits$ to the perceptions that are changing in between trials (i.e. $\mathcal{I}_{Maze6} \subseteq \{., O, F\}^L \cdot 0, 1^{ABits}$). The additional bits can be interpreted as further perceptions that are irrelevant for the maze task such as light, temperature, color, or sound. Considering a non-generalizing model learner it would be necessary to learn the environment $2^{|ABits|}$ times since it is not able to generalize over those irrelevant bits.

Although the ACS without GA is evolving a generalized environmental model it is trapped by those simple additional bits. The left-hand side of figure 12 reveals how the ACS without GA suffers from the additional bits in terms of the model size. Note that the population size in the case of $|ABits| = 4$ is divided by 50 and thus reaches after 50000 steps a size of 5000 classifiers and still increases. Also the model learning speed is slightly influenced by the bits. On the other hand, with the GA application we can observe a convergence of the population size to the one without GA and without additional bits even with 12 additional bits. Moreover, the model-building performance appears to be less influenced with the GA than without it. This shows that the ACS with GA is able to focus on the relevant attributes in the

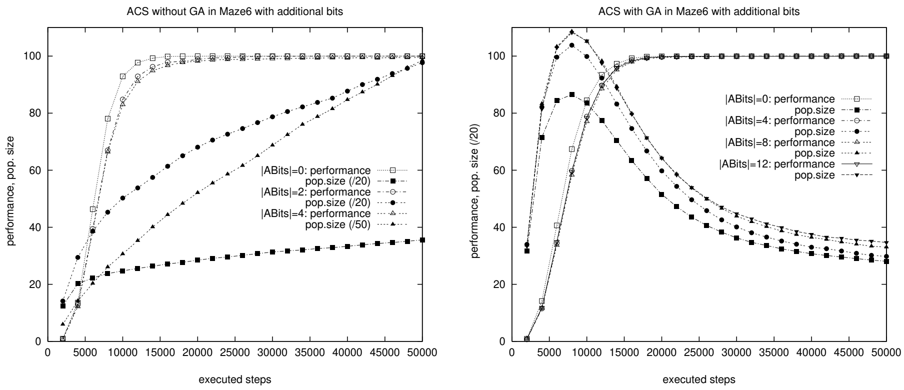

Figure 12. Adding additional bits in the Maze6 environment clearly shows that the GA is able to focus on the relevant attributes.

conditions and consequently forms a sort of attention mechanism in the conditions.

Furthermore, we investigated some parameter influences in Maze6. Model learning is fastest (without any additional mechanisms) when the probability of exploration ϵ is set to one since each state action transition is encountered as frequently as possible in the environment. However, when an adaptive behavior is desired from the beginning, ϵ cannot be set to one. Thus, we investigated the model learning performance of ACS with decreased ϵ. The main concern was whether if the GA destroys the model learning capability due to its randomized generalization. Often problems in learning have been reported when the probability of encountering each state-action pair was not equal. However, the left-hand side in figure 13 reveals that the GA is not influencing the performance in a negative way. As predicted, the model learning speed decreases with decreasing ϵ but it decreases in the case without GA similarly to the case with GA. This shows the relative robustness of the GA in environments with unequal experience distributions.

The right-hand side of figure 13 confirms the robustness of the θ_{ga} parameter in the GA. Although a more frequent GA application was anticipated to destroy the model learning capability at least in the range of $\theta_{ga} = 0$ to $\theta_{ga} = 200$ nearly no influence was observed in the model learning capabilities. The population size, as expected, increases at the beginning of a run with a higher GA threshold, since early in the run the ALP strongly generates offspring classifiers and the GA is not deleting useless offspring fast enough resulting in action set sizes that often are much bigger than θ_{as}. Later, the convergence with a lower GA frequency takes longer. A minor decrease in performance for a threshold

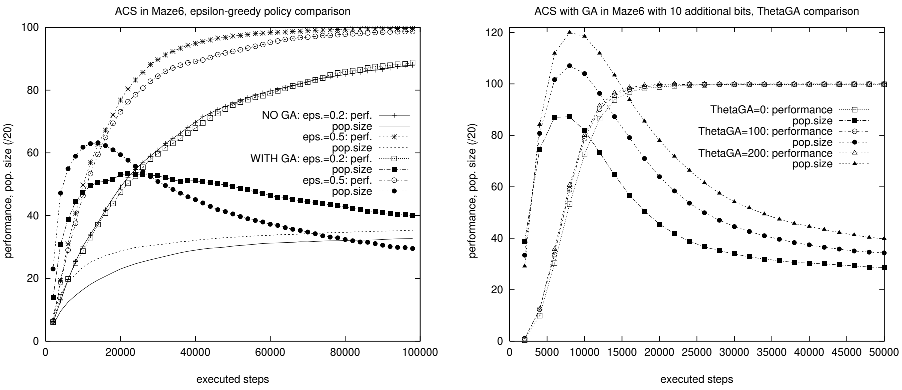

Figure 13. The comparison of the ACS with and without GA application on the left-hand side shows that the GA does not decrease the performance further when decreasing the exploration probability ϵ. The graph on the right-hand side shows that the success of the GA application is only slightly dependent on the value of the GA threshold θ_{ga}.

of zero is observable. Generally, this shows that the performance of the GA is influenced only slightly by the θ_{ga} threshold.

Model in Woods14 While in Maze6 the model-learning problem is of greater interest, Woods14 is better suited for a reward-learning study. Cliff and Ross (1994) studied the performance of ZCS (Wilson, 1994) in the environment. Although the problem appears to be trivial since there is only one path to the food, many problems can occur when trying to generalize. In the ACS, in terms of model learning the question arises if the frequently encountered movement against walls (due to the completely random action selection) actually causes trouble in the system. Especially since the ACS does not form any classifiers that predict no change after an action, the GA could be disruptive when applied in a set where the ALP actually does not cause any changes. However, the left graph in figure 14 reveals that the model with and without GA evolves similarly. The population again increases early in the run further when the GA is applied but converges to a similar size later.

Interestingly, experiments without the useless case and with no restriction in the crossover operator actually reveal possible disruption (right-hand side of figure 14). Adding either the useless case or the crossover-restriction or both (adding both is shown in the figure) solves this disruption. This indicates that if the crossover operator is unrestricted and combines conditions of changing and unchanging anticipations, ACS is not able to reach a complete knowledge reliably. Thus, a restriction in the crossover operator can prevent disruption.

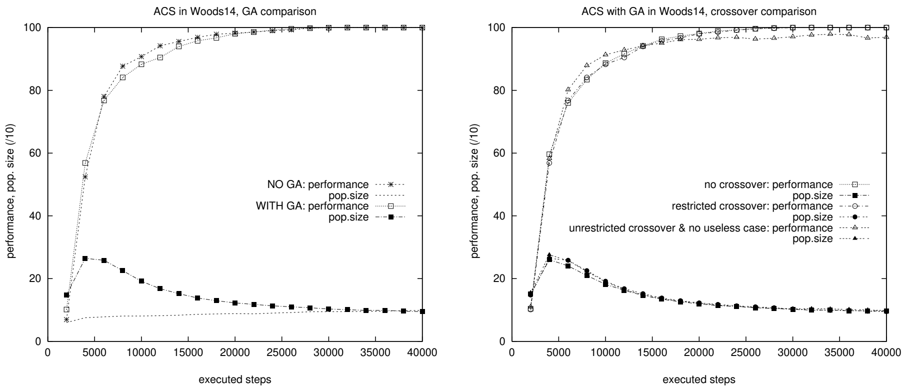

Figure 14. While the model learning performance is not influenced by the GA in Woods14, applying an unrestricted crossover operator in combination with no useless case application actually results in an influence.

Policy Learning

While the ACS is mainly a model learner the possibility to evolve a policy while forming the model is an important, beneficial feature. As we will discuss later, it is important that the evolving model be specific enough to allow an appropriate reinforcement mechanism. This property is given in Maze6 as well as in Woods14. The runs presented in this section alter between one exploration trial, where reinforcement learning and model building takes place, and one exploitation trial, where the model is purely exploited and only the reinforcement learning method is applied, identical to (Wilson, 1995). We plot population size and the average steps to food during exploitation mode in these graphs. Moreover, we restricted the length of one trial to fifty steps. Although this might have a slight effect on the policy learning capabilities (see Lanzi (1999) for influences in XCS) this restriction was made to provide an upper bound of one trial length. Doing that, it is not possible to 'hide' a huge number of actual steps in one trial. Also the change in population size allows a comparison to earlier curves and can reveal how many actual steps were executed so far approximately.

Policy in Maze6 Although Lanzi (1999) reported reward learning problems in XCS in Maze6, these problems are not observed in the ACS runs. In Maze6, we observe that the formation of an optimal policy is only slightly influenced by the GA application. The left-hand side of figure 15 reveals that it takes a little longer to reach an optimal performance when applying the GA. However, the difference in the population size is large. While the population size of the ACS without GA reaches a size of about 9000 after 4000 trials and still grows,

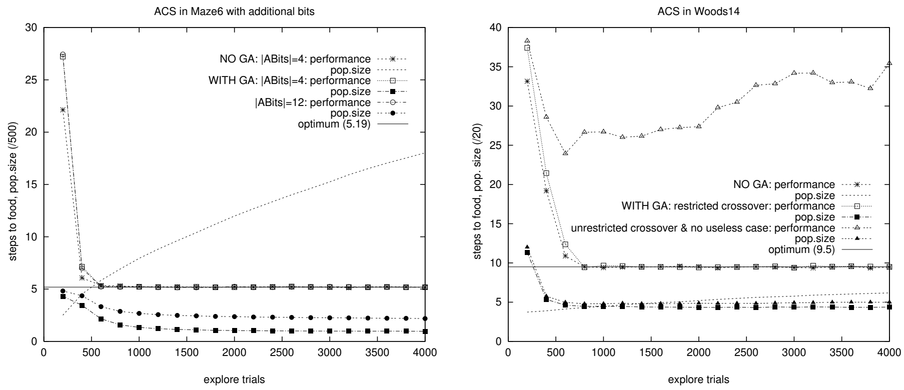

Figure 15. The GA does not influence the policy learning mechanism. Actually, it even allows a faster policy formation when adding additional bits in Maze6. However, if crossover is not restricted, the performance is effected.

the population size of the ACS with GA—even with twelve additional attributes in the perceptions—converges to about 500.

Policy in Woods14 Runs in Woods14 shown at the left-hand side of figure 15 show that the GA application only slightly decreases the learning of the policy. However, again—even without additional bits—a further decrease in the population size can be observed. Moreover, the curve with unrestricted crossover and no useless case application shows that crossover can also disrupt the policy learning capabilities. By contrast to the results reported in Butz et al. (2000b) ACS with GA now reaches an optimal number of steps without further parameter modifications. This is due to the different parameter initialization of offspring which now multiplies the quality of the offspring by 0.5 rather than setting the offspring's quality directly to 0.5.

New Challenges

The last section showed that the ACS is able to perform well in environments with diverse properties. The system is able to evolve an accurate, complete, and generalized environmental model online. However, many challenges remain:

1. The anticipatory learning mechanism represents only changes.

2. The reinforcement learning approach appears to be fragile.

3. A faster or more appropriate adaptation was not shown.

4. Environments with further memory requirements are not solvable.
5. The model learning speed depends on the amount of exploration.
6. The attention mechanism only works in the conditions.
7. Non-determinism provides further challenges.

Most of the challenges appear to be crucial also in the broader sense of anticipatory learning systems and even more generally adaptive learning systems. We discuss each of the points in somewhat more detail in the reminder of this section.

The Anticipatory Learning Mechanism

Right now the introduced ALP only considers changes as important and thus only evolves classifiers that represent such changes. A further enhancement for the explicit representation of non-changing pairs could be possible as well but was not pursued herein. In this matter, the identification or characterization of environmental properties would be important where a representation of non-changes is a prerequisite for an appropriate behavioral adaptation or the application of higher level processes such as planning.

The Reinforcement Learning Approach

Right now the evolving policy completely relies on the evolving model. Although both curves in figure 15 show that the GA does not disrupt the reinforcement learning mechanism this is because the evolving model is still sufficiently precise to enable appropriate reinforcement learning. Lanzi (2000b) puts this statement into a formal framework comparing the Q-learning approach with generalizing reinforcement learning mechanisms such as learning classifier systems. Essential for the ACS is that the conditions of all condition-action-effect representations don't become over-general in terms of the necessary condition-action-reward relation. Thus, in environments where the environmental model can be represented in a more generalized way than the reward model, the current ACS can get into trouble.

In other frameworks explicit observations of this kind have been made, for example by Whitehead and Ballard (1991). Their approach to the problem is to detect and explicitly lower the action values of oscillating rules—rules that specify such over-general states. In the learning classifier system field Grefenstette (1988) identified the problem of averaging in LCS which however was rather a defect of the fitness approach. Dorigo and Colombetti (1998) detected similar reward oscillation in their learning classifier system approach and solved the problem with a *MutSpec* operator directly specializing classifiers once such an oscillation is detected.

An investigation of such problems in ACS and essentially the realization of a stronger interaction of reinforcement learning and model learning is certainly necessary. One solution to the problem would be to modify the fitness measure in the ACS. While the fitness measure in the learning methods is mainly used by the genetic generalization, it also plays a crucial role in the model representation. A consideration of an appropriate combination of reinforcement and anticipatory fitness measure appears to be a new enterprise. Another approach would be to include a rather independent reward fitness measure which can influence the model formation on its own.

Adaptation

The above challenge seeks to improve the current policy learning process itself. Nevertheless, another challenge in the ACS is to begin with the exploitation of the currently evolving environmental model. First approaches in these terms were investigated in Stolzmann and Butz (2000) confirming the possibility of a speed up in the model formation by an online planning approach and in Stolzmann et al. (2000) applying cognitive methods to improve adaptation.

Essential in any model-exploitation approach appears to be the question for what the evolving model can or should be used. Which environmental properties lead to the necessity of including anticipations in the evolution of behavior? This question directly leads to the question of the usefulness and necessity of anticipations itself. Analysis and discussions can be found in Rosen (1991) and Holland (1992).

Moreover, the population-based approach promises faster adaptation to changing environments. Analysis for the ACS and also LCSs are necessary to confirm this prediction.

Memory

Many other adaptive approaches study the possible usage of additional memory. While the population of classifiers in itself actually represents some sort of memory a further memory representation becomes necessary in environments with perceptual aliasing states, which are part of the so called partially observable Markov decision processes (POMDPs). In ACS a first approach uses action chunking to solve small POMDP problems (Stolzmann, 2000).

While the traditional message list approach did only show limited success, a different memory register approach in XCS has been published in Lanzi and Wilson (2000) where XCS implicitly learns to set additional bits as indicators for the current aliasing state. However, Lanzi (2000a) showed that this approach does not guarantee a reliable solution and further analyzes the perceptual aliasing problem in a more unified view. Nonetheless, it appears to be necessary to define and detect when memory is necessary in an environment in general and how it could be involved in the behavioral process of ACS.

Exploration vs. Exploitation

As has been observed in the experiments above, the model-building speed suffers from the exploration-exploitation dilemma. This observation was made in several distinct learning systems.

Sutton and Barto (1998) provide the viewpoint from the reinforcement learning field. Moreover, they mention an approach in terms of model learning. Their Dyna-Q+ system gradually favors state-action pairs that were tested longer ago in the real environment resulting in a system behavior that tends to verify its environmental model. Because of the generalized, probabilistic transitions in ACS, an additional tendency that prefers uncertain regions in the state transition space could be useful.

More globally, Wilson (1995) discusses the dilemma in the XCS context. He proposes a dynamically controlled approach in that the reward prediction error measure in the classifiers could be an indicator for what amount of exploration is currently appropriate. The dilemma in the ACS, though, rather addresses model learning. Thus, the percentage of accurate anticipations or the quality of the classifiers would be the corresponding measure in the ACS.

Attention

One of the biggest challenge for the ACS is the solution of environments with many additional changing attributes. Or rather, environments where different subsets of perceptions are relevant in different situations and/or actions. The ALP in the ACS essentially has a *strong relevance assumption*. All encountered changes in an environment are basically believed to be cause by itself. Thus, it is extremely difficult to enable it to ignore unrelated changes. How is it possible to realize that an action can only manipulate parts of an environment and consequently ignore changes in other parts of an environment? Or, in a more general way, which situational and intentional properties determine the relevance of each detector in an environment?

This question directly relates to the diverse investigations of attentional processes and the formation of attentional spots regardless if in the visual system or other parts of a cognitive system (see LaBerge (1995) for a review with the direct relation to neurosciences). Although the evolving accurate, maximally general classifiers in the ACS realize something like an attention mechanism in the conditions, humans and also all higher animals are able to focus on the relevant changes in an environment, changes that are strongly related to themselves. The ACS currently is not able to ignore irrelevant changes. To define and identify relevance in the ACS and consequently create an anticipatory, attentional system appears to be a highly challenging endeavor.

Non-Determinism

An attentional process, however, is not sufficient to deal with non-determinism. As already explained above, it can still corrupt the specialization process. More severely though is the possible disruption in the anticipations. Due to probabilistic state transitions or detector information (i.e. noise) changes are not deterministically predictable anymore. A first approach to this problem is given in Butz et al. (2001). Nonetheless, the further examination and analysis of the problem is necessary. Moreover, it appears to be necessary to clarify the difficulty of such environments in terms of anticipations and model building.

While these scenarios address the problem of probabilistic changes or perceptions, an even more disruptive case could be generated by other forces in the environment such as other agents or natural forces e.g. wind or temperature. In this case the probabilities would further depend on the—possibly undetectable—environmental conditions. Thus, this problem appears to be a currently distant challenge for the ACS.

Summary and Conclusions

The aim of this paper was to clarify the functioning of the ACS and to introduce a genetic generalization pressure to the ACS in order to generate accurate and maximally general classifiers. After a broader look at the background of the system, the current system was introduced. First, we described how ACS is embedded in the LCS framework and how classifiers are augmented with an effect part enabling concrete anticipations. Furthermore, we outlined that this explicit representation of anticipations essentially allows the representation of an internal environmental model. The reinforcement learning component, realized by a modified Q-learning mechanism, was described after that. Next, each facet of the anticipatory learning process (ALP), the main learning method in the ACS, was explained. The ALP essentially evolves more specialized rules out of inaccurate, more general ones resulting in the formation of a complete environmental model. The explanations directly led to the observation of possible over-specializations in the model revealing the need for some kind of generalization. Due to the population-based approach and the present accuracy measure we consequently applied a genetic generalization process. After description of idea and method we investigated the interaction of the deterministic ALP and the stochastic genetic generalization. The results above empirically confirmed that the processes are able to work together for a common goal: the evolution of an accurate, maximally general world model.

While we believe that these results confirm the importance of the ACS in their own rights, we intended to stress important factors along those lines which we summarize now. First, we emphasized the importance of the accuracy measure in learning classifier systems (LCSs). While previous LCSs struggled with the problem of different reward levels, this problem is surmounted by the accuracy measure. Moreover, the accuracy approach results in the evolution of a complete environmental mapping rather than just a mapping of the apparent best. Second, we intended to highlight the advantages of LCSs. Despite its apparent restrictions in expressiveness the crucial factors are the population-based approach allowing diversity and promising faster adaptivity as well as the readability of the acquired knowledge enabling explicit knowledge recovery and understanding. Third, research in the ACS relates directly to research in anticipatory systems. As we have, the presence of anticipations appears to be a crucial factor in adaptive behavior and even life itself. Observations of anticipatory behavior in higher animals support this statement. However, it is not clear to what extend anticipations are necessary for an appropriate behavior and in what way anticipations are related to other behavioral characteristics.

The three factors are comprised in the ACS. Its classifier quality measure realizes the accuracy measure. The population-based approach with classifiers as individuals corresponds to the LCS idea. The three-part rule approach with its condition, action, and effect part codes direct anticipations in the system.

Thus, the ACS provides a suitable framework to explore the utility and necessity of anticipations in further detail. Due to the complete model formation it appears to be possible—at least in deterministic environments—to exploit the model or simply examine the evolving model and detect important patterns in diverse environments. Although the restriction to determinism discourages applications in real-world environments, we encourage the examination and use of the ACS in simplified real-world problems. Insights in this settings can reveal important further limits as well as possible or necessary extensions for the realization of various cognitive mechanisms and a competent adaptive behavior.

Work on the ACS and research in anticipatory online generalizing systems, in general, is just in its beginning. We hope that our introduction of the ACS, the encouraging model learning results, and the discussed relations to other ideas and systems are helpful to many related lines of research.

Acknowledgments

The work was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant F49620-00-0163. Research funding for this work was also provided by a grant from the National Science Foundation under grant DMI-9908252. The US Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon. Additionally, the work was supported by the German Research Foundation (DFG) under grant HO1301/4.

The views and conclusions contained herein are those of the authors 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, or the U.S. Government.

References

Booker, L. B., D. E. Goldberg, and J. H. Holland: 1989, 'Classifier Systems and Genetic Algorithms'. *Artificial Intelligence* **40**, 235–282.

Butz, M. V., D. E. Goldberg, and W. Stolzmann: 2000a, 'Introducing a genetic generalization pressure to the Anticipatory Classifier System: Part 1 - Theoretical Approach'. In: D. Whitely, D. E. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, and H.-G. Beyer (eds.): *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*. San Francisco, CA, pp. 34–41, Morgan Kaufmann.

Butz, M. V., D. E. Goldberg, and W. Stolzmann: 2000b, 'Introducing a genetic generalization pressure to the Anticipatory Classifier System: Part 2 - Performance Analysis'. In: D. Whitely, D. E. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, and H.-G. Beyer (eds.): *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*. San Francisco, CA, pp. 42–49, Morgan Kaufmann.

Butz, M. V., D. E. Goldberg, and W. Stolzmann: 2000c, 'Investigating Genetic Generalization in the Anticipatory Classifier System'. In: M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merely, and H.-P. Schwefel (eds.): *Parallel Problem Solving from Nature - PPSN VI, LNCS 1917*. Berlin, pp. 735–744, Springer-Verlag.

Butz, M. V., D. E. Goldberg, and W. Stolzmann: 2001, 'Probability-Enhanced Predictions in the Anticipatory Classifier System'. In: P. L. Lanzi, W. Stolzmann, and S. W. Wilson (eds.): *Advances in Learning Classifier Systems - Third International Workshop, IWLCS-200, LNAI 1996*. Berlin, pp. 38–52, Springer-Verlag. in press.

Cliff, D. and S. Ross: 1994, 'Adding Temporary memory to ZCS'. *Adaptive Behavior* **3**(2), 101–150.

Dorigo, M. and M. Colombetti: 1998, *Robot Shaping, an experiment in behavior engineering*, Intelligent Robotics and Autonomous Agents. Cambridge, MA: MIT press.

Drescher, G. L.: 1991, *Made-up minds, a constructivist approach to artificial intelligence*. Cambridge, MA: MIT Press.

Goldberg, D. and J. Richardson: 1987, 'Genetic algorithms with sharing for mulitmodal function optimization'. In: J. J. Grefenstette (ed.): *Proceedings of the Second International Conference on Genetic Algorithms*. Hillsdale, NJ, pp. 41–49, Lawrence Erlbaum Associates.

Goldberg, D. E.: 2002, *The design of innovation: Lesssons from and for competent genetic algorithms*, Genetic Algorithms and Evolutionary Computation. Boston, MA: Kluwer Academic Publishers.

Goldberg, D. E. and K. Deb: 1991, 'A comparative analysis of selection schemes used in genetic algorithms'. *Foundations of Genetic Algorithms* **1**, 69–93. (Also TCGA Report 90007).

Grefenstette, J. J.: 1988, 'Credit Assignment in Rule Discovery Systems Based on Genetic Algorithms'. *Machine Learning* **3**, 225–245.

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

Holland, J. H.: 1985, 'Properties of the bucket brigade algorithm'. In: *Proceedings of an international conference on genetic algorithms and their applications*. Carnegie-Mellon University, Pittsburgh, PA, pp. 1–7, John J. Grefenstette.

Holland, J. H.: 1990, 'Concerning the emergence of tag-mediated lookahead in classifier systems'. In: S. Forrest (ed.): *Emergent Computation*. North-Holland, pp. 188–201.

Holland, J. H.: 1992, *Adaptation in Natural and Artificial Systems : An Introductory Analysis With Applications to Biology, Control, and Artificial Intelligence (Complex A)*. MIT press.

Holland, J. H. and J. S. Reitman: 1978, 'Cognitive systems based on adaptive algorithms'. In: D. A. Waterman and F. Hayes-Roth (eds.): *Pattern Directed Inference Systems*. New York: Academic Press, pp. 313–329.

James, W.: 1981 (orig.1890), *The Principles of Psychology (Vol.2)*. Cambridge, MA: Harvard University Press.

Kovacs, T.: 1996, 'Evolving Optimal Populations with XCS Classifier Systems'. Master's thesis, School of Computer Science, University of Birmingham, Birmingham, U.K.

Kovacs, T.: 1997, 'XCS Classifier System Reliably Evolves Accurate, Complete, and Minimal Representations for Boolean Functions.'. In: Roy, Chawdhry, and Pant (eds.): *Soft Computing in Engineering Design and Manufacturing*. pp. 59–68, Springer-Verlag, London.

Kuvayev, L. and R. S. Sutton: 1996, 'Model-based reinforcement learning with an approximate, learned model'. In: *Proceedings of the Ninth Yale Workshop on Adaptive and Learning Systems*. New Have, CT, pp. 101–105.

LaBerge, D.: 1995, *Attentional processing, the brain's art of mindfulness.* Harvard University Press, Cambridge, MA.

Lanzi, P. L.: 1999, 'An Analysis of Generalization in the XCS Classifier System'. *Evolutionary Computation* **7**(2), 125–149.

Lanzi, P. L.: 2000a, 'Adaptive agents with reinforcemen learning and internal memory'. In: J.-A. Meyer, A. Berthoz, D. Floreano, H. Roitblat, and S. W. Wilson (eds.): *From Animals to Animats 6: Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior.* Cambridge, MA, pp. 333–342, MIT Press.

Lanzi, P. L.: 2000b, 'Learning Classifier Systems from a Reinforcement Learning Perspective'. Technical Report 00-03, Dipartimento di Elettronica e Informazione, Politecnico di Milano.

Lanzi, P. L. and S. W. Wilson: 2000, 'Toward optimal classifier system performance in non-Markov environments'. *Evoultionary Computation* **8**(4), 393–418.

Riolo, R. L.: 1991, 'Lookahead planning and latent learning in a classifier system'. In: J.-A. Meyer and S. W. Wilson (eds.): *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior.* Cambridge, MA, pp. 316–326, MIT Press.

Rosen, R.: 1985, *Anticipatory Systems.* Oxford, UK: Pergamon.

Rosen, R.: 1991, *Life Itself (Complexity in Ecological Systems Series).* New York, USA: Columbia University Press.

Stolzmann, W.: 1997, *Antizipative Classifier Systeme [Anticipatory classifier systems].* Osnabrueck, Germany: Shaker Verlag, Aachen, Germany.

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

Stolzmann, W.: 2000, 'An Introduction to Anticipatory Classifier Systems'. In: P. L. Lanzi, W. Stolzmann, and S. W. Wilson (eds.): *Learning Classifier Systems: From Foundations to Applications, LNAI 1813.* Berlin, pp. 175–194, Springer-Verlag.

Stolzmann, W. and M. V. Butz: 2000, 'Latent learning and action-planning in robots with Anticipatory Classifier Systems'. In: P. L. Lanzi, W. Stolzmann, and S. W. Wilson (eds.): *Learning Classifier Systems: From Foundations to Applications, LNAI 1813.* Berlin, pp. 301–317, Springer-Verlag.

Stolzmann, W., M. V. Butz, J. Hoffmann, and D. E. Goldberg: 2000, 'First Cognitive Capabilities in the Anticipatory Classifier System'. In: J.-A. Meyer, A. Berthoz, D. Floreano, H. Roitblat, and S. W. Wilson (eds.): *From Animals to Animats 6: Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior.* Cambridge, MA, pp. 287–296, MIT press.

Sutton, R. S.: 1991a, 'DYNA, an integrated architecture for learning, planning, and reacting'. In: *Working Notes of the 1991 AAAI Spring Symposium on Integrated Intelligent Architectures.* pp. 151–155.

Sutton, R. S.: 1991b, 'Reinforcement Learning Architectures for animats'. In: J.-A. Meyer and S. W. Wilson (eds.): *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior.* Cambridge, MA, pp. 288–296, MIT Press.

Sutton, R. S. and A. G. Barto: 1998, *Reinforcement Learning: An Introduction.* Cambridge, MA: MIT Press.

Thislethwaite, D.: 1951, 'A critical review of latent learning and related experiments'. *Psychological Bulletin* **48**(2), 97–129.

Tolman, E. C.: 1932, *Purposive Behavior in animals and men.* New York: Appleton.

Tomlinson, A. and L. Bull: 2001, 'Symbiogenesis in Learning Classifier Systems: Corporations, Niches and Persistence'. *Artificial Life*. in press.

Watkins, C. J. C. H. and P. Dayan: 1992, 'Q-learning'. *Machine Learning* **8**(3), 272–292.

Whitehead, S. D. and D. H. Ballard: 1991, 'Learning to perceive and act'. *Machine Learning* **7**(1), 45–83.

Widrow, B. and M. Hoff: 1960, 'Adaptive switching circuits'. *Western Electronic Show and Convention* **4**, 96–104.

Wilson, S. W.: 1994, 'ZCS: A zeroth level classifier system'. *Evolutionary Computation* **2**(1), 1–18.

Wilson, S. W.: 1995, 'Classifier Fitness Based on Accuracy'. *Evolutionary Computation* **3**(2), 149–175.

Wilson, S. W.: 1998, 'Generalization in the XCS classifier system'. In: J. R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba, and R. Riolo (eds.): *Genetic Programming 1998: Proceedings of the Third Annual Conference*. San Francisco, pp. 665–674, Morgan Kaufmann.

Wilson, S. W.: 2000, 'Get real! XCS with continuous-valued inputs'. In: P. L. Lanzi, W. Stolzmann, and S. W. Wilson (eds.): *Learning Classifier Systems: From Foundations to Applications, LNAI 1813*. Berlin Heidelberg, pp. 209–219, Springer-Verlag.

Wilson, S. W. and D. E. Goldberg: 1989, 'A critical review of classifier systems'. In: J. D. Schaffer (ed.): *Proceedings of the Third International Conference on Genetic Algorithms*. San Mateo, CA, pp. 244–255, Morgan Kaufmann.