

**Sequential Problems that Challenge
Generalization in Classifier Systems**

Martin V. Butz & Pier Luca Lanzi
IlliGAL Report No. 2008007
March, 2008

Illinois Genetic Algorithms Laboratory
University of Illinois at Urbana-Champaign
117 Transportation Building
104 S. Mathews Avenue Urbana, IL 61801
Office: (217) 333-2346
Fax: (217) 244-5705

Sequential Problems that Challenge Generalization in Classifier Systems

Martin V. Butz & Pier Luca Lanzi^{†*}
University of Würzburg, Röntgenring 11, 97070 Würzburg, Germany
`butz@psychologie.uni-wuerzburg.de`

[†]Illinois Genetic Algorithm Laboratory (IlliGAL)
University of Illinois at Urbana Champaign, Urbana, IL 61801, USA

*Dipartimento di Elettronica e Informazione
Politecnico di Milano. P.za L. da Vinci 32, I-20133, Milano, Italy
`pierluca.lanzi@polimi.it`

March 20, 2008

Abstract

We present an approach to build sequential decision making problems which can challenge the generalization capabilities of classifier systems. The approach can be applied to any sequential problem defined over a binary domain and it generates a new problem with bounded sequential difficulty and bounded generalization difficulty. As an example, the approach is used here to generate two problems with a simple sequential structure, huge number of states (more than a million), and many generalizations. These problems are used to compare a classifier system with effective generalization (XCS) and a learner without generalization (Q-learning). The experimental results confirm what was previously found mainly using single-step problems, also in sequential problems with huge state spaces, XCS can generalize effectively by detecting those context-dependent structures that are necessary for optimal sequential behavior.

1 Introduction

Generalization in learning classifier systems has been widely studied using classification problems [19, 7, 3, 4, 20], but rarely using sequential problems [19, 8]. The vast majority of the sequential problems introduced in the literature are characterized by a simple, although challenging, sequential structure which is usually described by few (usually less than a hundred) states [2, 11, 1, 17]. Most important, they usually allow few generalizations [14, 15] and therefore, since they do not challenge the generalization capabilities of classifier systems, they also make the advantage of effective generalization risible. As a consequence, the study of generalization in classifier systems focuses on classification problems that currently represent the best testbed for generalization.

An early example of effective generalization in a sequential problem was provided by Wilson in [19], where it was shown that XCS [18] could evolve just 35 classifiers to solve a problem, namely `Woods2`, which would require 560 classifiers without generalization. Noticeably, to evolve those 35 classifiers, XCS required a population of 800 classifiers—the population required to achieve accurate generalization was actually larger than the one required without generalization (although the population size of distinct classifiers in XCS stayed on less than 560 classifiers, that is, about

500 classifiers throughout the run). A stronger generalization example was provided by Butz et al. [8], who added 30 irrelevant bits to a simple sequential problem and showed that XCS could easily get rid of the irrelevant inputs through extensive generalizations. In this case, the state space was increased by a factor of 2^{30} whereas the population size necessary to solve the problem had to be less than doubled. Further generalization capabilities were confirmed in the XCS component of XACS in a blocks-world problem [5] as well as with even up to 90 irrelevant bits in more challenging sequential environments [7]. Except for in the blocks-world problem example, though, in all these tasks the learning mechanism had to simply learn to ignore certain bits in the general case and not context-dependent.

This observation suggests that the sequential problems currently available in the literature do not provide an adequate ground to study generalization. To study generalization in the general case, we need to create problems (a) with a simple structure, so that the sequential behavior can be easily analyzed; (b) consisting of a large (possible, huge) number of states, so that generalization becomes mandatory; (c) allowing many generalizations, so that the advantage provided by learning classifier systems can be clearly demonstrated and measured.

In this paper, we propose a simple procedure to build sequential problems that satisfy these three requirements which can be applied to any sequential problem defined over a binary domain. As an example, we use this procedure to generate two testbeds with a simple sequential structure, huge number of states (more than a million), and many generalizations. The two problems allow a *fair*—to our knowledge the first fair—comparison between a learner with no generalization and a classifier system with powerful generalization on two dimensional grid (maze-like) environments.

2 Sequential Problems that Challenge Generalization

To build sequential problems that can challenge the generalization capabilities of classifier systems we apply what Butz et al. [9] did for Boolean functions. The approach is extremely simple and it can be applied to any sequential problem defined over a binary domain. It basically consists in taking a sequential binary problem and replacing each input bit with the argument of a Boolean function that computes the corresponding bit. This process generates a sequential problem that is more challenging than the original one in two respects. Firstly, it is defined over a much larger problem space so that effective generalization becomes mandatory. Secondly, its input space has a hierarchical structure in that the high level sequential behavior is determined through a computation performed on the low-level inputs. Thus, to be effective, the generalization mechanism must also identify which groups of inputs actually determine the high level sequential behavior of the system. To illustrate our approach, we apply it to two simple problems taken from the well-known woods environments, i.e., **Woods1** and **Maze4** (Figure 1), using Boolean parity.

Woods environments [18] are mazes containing obstacles (represented by “T” symbols), goals (represented by “F” symbols), and empty positions. The system can occupy any of the empty positions and it is allowed to move in any adjacent position that is empty; it has eight sensors, one for each adjacent position. Sensors are encoded with two bits: obstacles are encoded with 10, goals with 11, empty positions with 00; thus the system’s input is a strings with 16 bits (2 bits \times 8 positions). There are eight possible actions, one for each possible adjacent position, that are encoded using three bits. The system must learn how to reach the goal from any empty position; when the goal is reached, the problem ends and the system receives a constant reward of 1000; otherwise, it will receive zero. **Woods1** (Figure 1a) is a toroidal maze consisting of 16 empty positions (i.e., 16 distinct states) and one goal. **Maze4** (Figure 1b) is a maze consisting of 26 empty positions (i.e., 26 distinct binary states) and one goal.

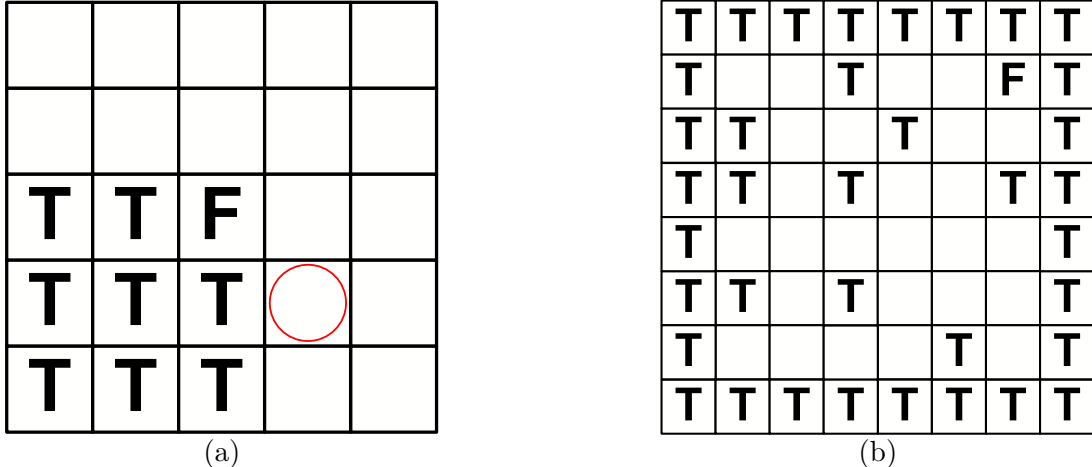


Figure 1: Two simple sequential problems: (a) Woods1, (b) Maze4.

To build a more challenging version of woods environments, we change the coding of states by replacing each input bit with the argument of a Boolean function that is used to compute the same input. For instance, we can use the Woods1 environment in Figure 1a to build a hierarchical version of it, $\text{Woods1}(\text{par}_2)$, using a *two-bits parity* function as follows. When the system enters an empty position of $\text{Woods1}(\text{par}_2)$, the input configuration is initially computed as usual [18]: for instance, when the system is in the position highlighted with a circle in Figure 1a, the input configuration is computed as “0000000000101011”. Then, each input bit is replaced by an argument of a *two-bits parity* function which can be used to compute the corresponding input bit: a zero is randomly mapped either into 00 or 11; a one is randomly mapped either into 01 or 10. For instance, the original input string “0000000000101011” can be mapped either to “0011110110011111100011110001001” or to “0000001111111111111010001000110”. This procedure transforms the original Woods1, consisting of 16 states, into $\text{Woods1}(\text{par}_2)$ which consists of 2^{20} states and without generalization would require 8388608 classifiers (one classifier for each state-action pair with 1048576 states and 8 actions), but has the *same* sequential structure of Woods1 and, most important, it is still Markov [16].

3 Demonstrating Effective Generalization

We compared a classifier system with effective generalization, XCS [18], with a learner without generalization, Q-learning [16], using $\text{Woods1}(\text{par}_2)$ and $\text{Maze4}(\text{par}_2)$. For this purpose, we followed the usual experimental design and we used the typical parameter settings [18]: $\beta = 0.2$; $\gamma = 0.7$; $\alpha = 0.1$; $\epsilon_0 = 10$; $\nu = 5$; $P_{\#} = 1.0$; $\chi = 0.8$, $\mu = 0.01$, $\theta_{del} = 20$; $\theta_{GA} = 25$; $\delta = 0.1$; GA-subsumption is on in Woods1 and it is off in Maze4, as usual; action-set subsumption is off. Q-learning [16] is applied with the same β and γ as XCS, the initial Q-table is initialized with zero values.

Figure 2a compares the performance of Q-learning with that of XCS using a population of 1600 classifiers in Woods1. As can be noted, XCS is slower than Q-learning: the problem has only 16 states, thus generalization does not provide any advantage. XCS also evolves larger solutions. The final populations evolved by XCS contain an average of 390 classifiers whereas a fully specific solution would require just 128 classifiers (or equivalently, a Q-table with 128 entries). However, when the number of states involved is much larger the results are much different. Figure 2b compares the performance of Q-learning and XCS using a population of 20000 classifiers in $\text{Woods1}(\text{par}_2)$; since

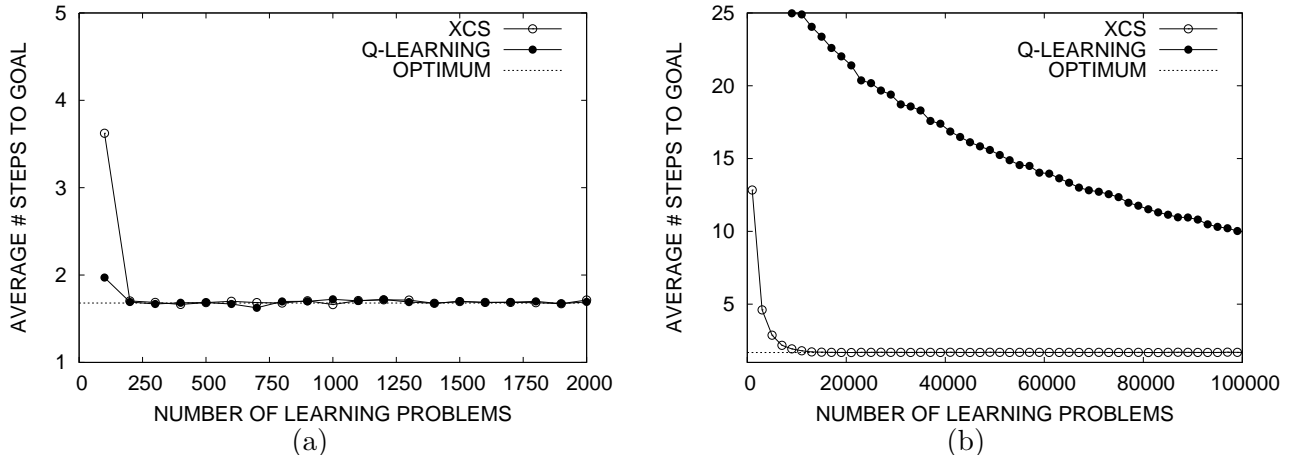


Figure 2: Performance of XCS and Q-learning in (a) Woods1 and (b) Woods1(par₂).

Woods1(par₂) has the same sequential structure of Woods1, both problems have the same optimum, i.e., notwithstanding the different scales, the optimum in Figure 2a is the same as Figure 2b. The plots show that the effective generalization in XCS increases the learning speed. XCS is much faster than Q-learning while evolving more compact solutions: XCS reaches optimal performance after 15000 learning problems, whereas Q-learning reaches optimality around 400000 steps; XCS evolves solutions containing an average of 4000 classifiers, whereas Q-learning requires a table with 8388608 entries, i.e., XCS without generalization would need 8388608 classifiers. Noticeably, since the problems have the same structure, the generalization process works similarly in the two cases. Figure 3 compares the evolution of the classifier populations in Woods1 (Figure 3a) and Woods1(par₂) (Figure 3b). Initially, when the solution space must be explored XCS generates many classifiers and, since the problem structure is similar, the amount of exploration is also similar: in both cases, the number of classifiers in the populations increases until the 70%. Then, when accurate generalizations are discovered, the populations shrink until they both converge to solutions which on the average consist of 20% of classifiers.

We repeated the same comparison using Maze4(par₂) which consists of 1703936 states—each one of the 26 original states generate 2^{16} states; the parameter setting is the same we used in the previous experiment except for the population size which in this experiment consisted of 120000 classifiers. Maze4(par₂) has the same sequential structure of Maze4 accordingly, as already shown in Maze4 [14, 15], XCS needs larger populations to reach optimal performance when no additional operators are employed to limit overgeneralization. Figure 4a compares the performance of Q-learning (solid dots) with that of XCS (empty dots). XCS rapidly reaches near optimal performance: when 50000 problems have been solved the average performance of XCS over the last 5000 problems is 3.7 while optimal performance is 3.5. Then, the learning process slows down until full optimality is reach around 100000 problem. This behavior is coherent with results discussed in [14, 15] for Maze4 and other sequential problems [10]. XCS quickly learns how to behave in the positions near the goal, but it needs more time for the positions farther from the goal since these are explored much less frequently [10] and they are more likely to cause overgeneralization [15]. Without generalization the learning is much slower: after 400000 learning problems Q-learning has an average performance of 9.8.

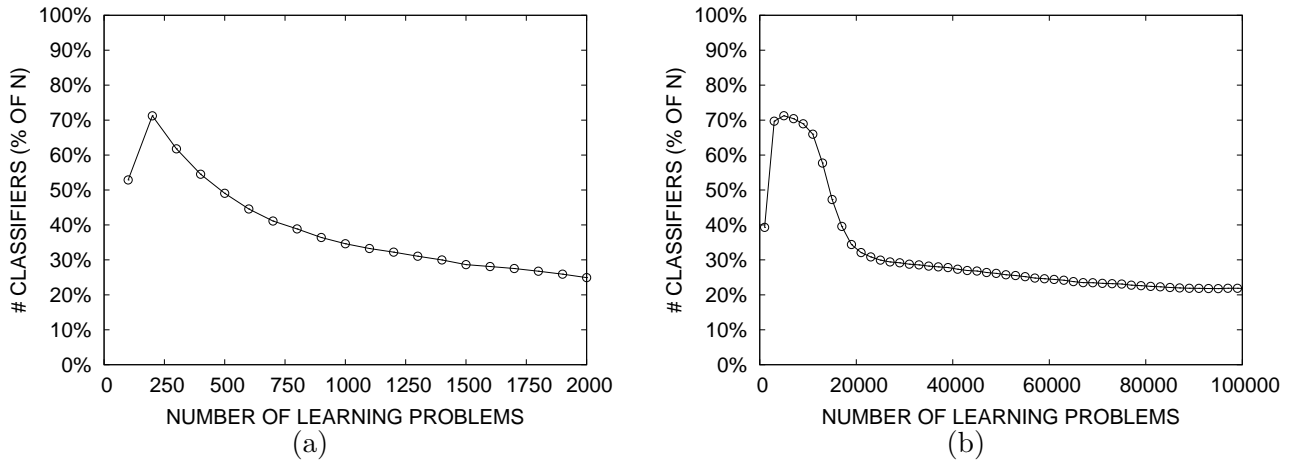


Figure 3: Number of classifiers in the population for XCS in (a) Woods1 and (b) Woods1(par₂).

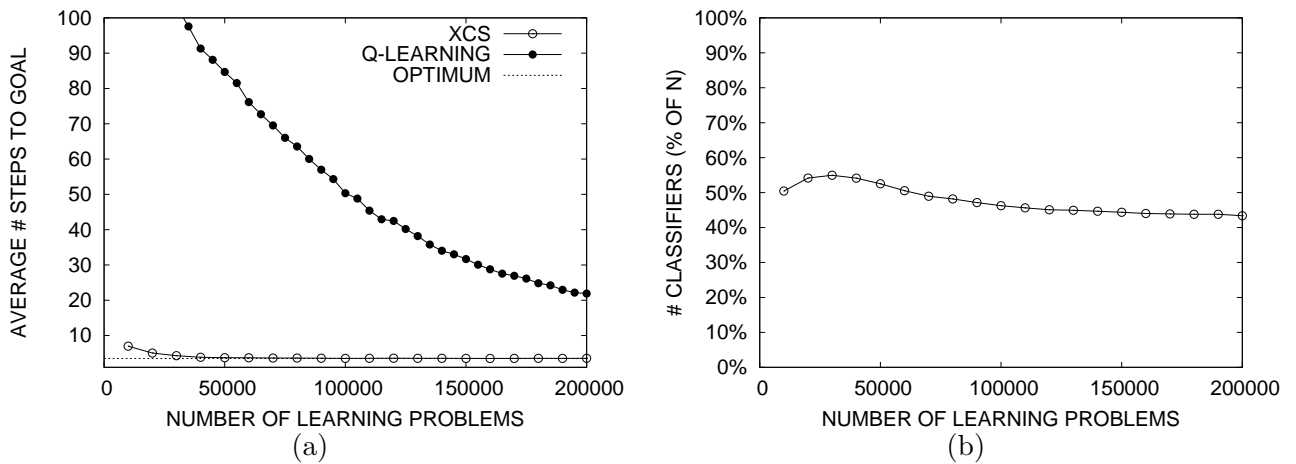


Figure 4: XCS and Q-learning applied to Maze4(par₂): (a) performance; (b) number of classifiers in the population.

4 Where From Here?

The research regarding sequential decision making with classifier systems has been mainly focused on the evolution and maintenance of optimal behavior and it ignored the issue of effective generalization (e.g., [2, 15, 6]). In the literature, there are very few sequential problems with challenging generalizations and the typical approach to build new ones has been to add irrelevant inputs [8, 5, 7]. However, such generalizations are context-independent and easy to evolve in that a wrong generalization of the irrelevant bits has no consequence on the performance but only the original inputs are relevant.

In this paper, we introduced an approach to build challenging sequential decision making problems, with context-dependent generalizations, from any existing sequential problem defined over a binary domain. The approach allows us to build problems of bounded difficulty both in terms of sequential behavior, since the new problem has the same sequential structure of the original one, and in terms of generalization, since by changing the underlying Boolean function we can vary the difficulty of the generalization. We also applied the approach to two well-known problems and showed that XCS can also generalize context-dependently, focusing on those attributes that are necessary to form accurate, action-dependent reward predictions. In effect, we showed that XCS can solve problems that would require more than 8 million tabular entries for an accurate solution representation with only 20000 classifiers, boiling down to 4000 classifiers at the end of a run.

The results confirm what already was shown in *much* simpler sequential problems [19] and in Boolean functions: XCS [18, 19, 7] can generalize effectively. While this was somewhat already known [19], it is important that we are now able to show that XCS is also able to generalize effectively in sequential decision problems beyond learning to simply ignore certain input attributes. Moreover, we showed that this generalization capability extended to an overall advantage, that is, the maximal population size was chosen way below the size requirement for an actual tabular representation. This indicates that XCS is able to detect context-dependently those relevant local structures that are necessary for the formation of accurate predictions. The results once again confirm the strong generalization capabilities of XCS and extend the knowledge confirming XCS's context-dependent generalization capabilities in multi-step problems. Back in the 1970s, when John Holland introduced learning classifier systems [12, 13], he had cognitive systems in mind that are able to detect that sensory information that is relevant for the development for efficient behavioral patterns. The work herein confirms that the current state-of-the art LCS system XCS can indeed be applied in such a cognitive system framework and is able to identify those perceptual cues that are relevant for the formation of accurate predictions. In the context of sequential decision making, in which the system strives to learn generalized payoff landscapes, XCS identifies those perceptual cues that allow accurate reward predictions. Thus, XCS realizes perception for action, a cognitive function that has been ignored for too long and seems to become increasingly important in the development of scalable, efficient, autonomous artificial cognitive systems.

References

- [1] A. J. Bagnall and Z. V. Zatuchna. On the classification of maze problems. In L. Bull and T. Kovacs, editors, *Foundations of Learning Classifier Systems*, volume 183 of *Studies in Fuzziness and Soft Computing*, pages 307–316. Springer, 2005.
- [2] A. M. Barry. The stability of long action chains in xcs. *Journal of Soft Computing*, 6(3–4):183–199, 2002.

- [3] L. Bull. *Applications of Learning Classifier Systems*. Studies in Fuzziness & Soft Computing. Springer-Verlag, 2004.
- [4] L. Bull, E. B. Mansilla, and J. H. Holmes. *Learning Classifier Systems in Data Mining*. Studies in Computational Intelligence. Springer-Verlag, 2008.
- [5] M. Butz and D. E. Goldberg. Generalized state values in an anticipatory learning classifier system. In M. Butz, O. Sigaud, and P. Gérard, editors, *Anticipatory Behavior in Adaptive Learning Systems*, volume 2684 of *Lecture Notes in Computer Science*, pages 282–301. Springer, 2003.
- [6] M. V. Butz. *Anticipatory Learning Classifier Systems*, volume 4 of *Genetic Algorithms and Evolutionary Computation*. Springer-Verlag, 2000.
- [7] M. V. Butz. *Rule-Based Evolutionary Online Learning Systems: A Principled Approach to LCS Analysis and Design*. Studies in Fuzziness and Soft Computing. Springer Verlag, Berlin-Heidelberg, Germany, 2006.
- [8] M. V. Butz, D. E. Goldberg, and P. L. Lanzi. Gradient descent methods in learning classifier systems: Improving xcs performance in multistep problems. *IEEE Transaction on Evolutionary Computation*, 9(5):452–473, Oct. 2005.
- [9] M. V. Butz, M. Pelikan, X. Llorá, and D. E. Goldberg. Automated global structure extraction for effective local building block processing in XCS. *Evolutionary Computation*, 14:345–380, 2006.
- [10] D. Cliff and S. Ross. Adding Temporary Memory to ZCS. Technical Report CSRP347, School of Cognitive and Computing Sciences, University of Sussex, 1995. <ftp://ftp.cogs.susx.ac.uk/pub/reports/csrp/csrp347.ps.Z>.
- [11] J. Drugowitsch and A. M. Barry. Xcs with eligibility traces. In H.-G. Beyer, U.-M. O’Reilly, D. V. Arnold, W. Banzhaf, C. Blum, E. W. Bonabeau, E. Cantu-Paz, D. Dasgupta, K. Deb, J. A. Foster, E. D. de Jong, H. Lipson, X. Llorá, S. Mancoridis, M. Pelikan, G. R. Raidl, T. Soule, A. M. Tyrrell, J.-P. Watson, and E. Zitzler, editors, *GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation*, volume 2, pages 1851–1858, Washington DC, USA, 25-29 June 2005. ACM Press.
- [12] J. H. Holland. Adaptation. In R. Rosen and F. Snell, editors, *Progress in Theoretical Biology*, volume 4, pages 263–293. Academic Press, New York, 1976.
- [13] J. H. Holland and J. S. Reitman. Cognitive systems based on adaptive algorithms, 1978. Reprinted in: *Evolutionary Computation. The Fossil Record*. David B. Fogel (Ed.) IEEE Press, 1998. ISBN: 0-7803-3481-7.
- [14] P. L. Lanzi. A Study on the Generalization Capabilities of XCS. In T. Baeck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms, April 19–23 East Lansing (MI)*, pages 418–425, San Francisco, July 1997. Morgan Kaufmann.
- [15] P. L. Lanzi. An Analysis of Generalization in the XCS Classifier System. *Evolutionary Computation Journal*, 7(2):125–149, 1999.
- [16] R. S. Sutton and A. G. Barto. *Reinforcement Learning – An Introduction*. MIT Press, 1998.

- [17] S. W. Wilson. ZCS: A zeroth level classifier system. *Evolutionary Computation*, 2(1):1–18, 1994. <http://prediction-dynamics.com/>.
- [18] S. W. Wilson. Classifier Fitness Based on Accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.
- [19] S. W. Wilson. Generalization in the XCS classifier system. In J. R. Koza, W. Banzhaf, K. Chellapilla, K. D. M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. G. H. Iba, and R. Riolo, editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 665–674. Morgan Kaufmann, 1998. <http://prediction-dynamics.com/>.
- [20] S. W. Wilson. Mining Oblique Data with XCS. Technical Report 2000028, University of Illinois at Urbana-Champaign, 2000.