An Introduction to Learning Classifier Systems

Tim Kovacs
Department of Computer Science
University of Bristol
kovacs@cs.bris.ac.uk
http://www.cs.bris.ac.uk/~kovacs

I Types of Problems

II Introduction to Classifier Systems

III Representation

IV Credit Assignment

V Rule Discovery

VI Conclusion

A Pattern Classification Problem

- Find simple rules which accurately classify mushrooms as either poisonous or edible.

One possible representation

- Use if-then rules to make classification decisions:

IF	small **AND** green	**THEN**	edible
IF	(small **AND** green) **OR** has-spots	**THEN**	poisonous
IF	small **AND** pink	**THEN**	poisonous
IF	small **AND** pink	**THEN**	edible
⋮			

- This seems a fundamental and simple approach to representing information.

A Reinforcement Learning Problem

- Given sensors and effectors, provide a self-adapting control system for a simulated frog, using only numerical feedback on actions.

One possible representation

- Use if-then rules to code control system:

IF	small **AND** dark **AND** buzzes	**THEN**	eat-it
IF	large **AND** white **AND** has-beak	**THEN**	hide
⋮			

Supervised Learning: "Learning from a teacher"

- Feedback to the learner consists of the correct answer, so the teacher needs a training set of correct examples.

```
Trial 1: Teacher: Does picture 1 show a car or a flower?
         Learner: A flower.
         Teacher: It's a car.

Trial 2: Teacher: Does picture 2 show a car or a flower?
         Learner: A flower.
         Teacher: It's a flower.
         ...
```

Reinforcement Learning: "Trial and error learning"

- Feedback is a reward (a number) representing the desirability of the action taken. The environment must provide a *reward function*.

```
Trial 1: Environment: You are in state 1.
         Learner: I'll take action A.
         Environment: Your reward is 100.

Trial 2: Environment: You are in state 9.
         Learner: I'll take action A.
         Environment: Your reward is 200.
         ...
```

Exploration and Exploitation

Did the learner get as much reward as it might have?

- To find out, it must revisit the same states and try (explore) other actions.

How much should it explore?

- The more it explores the more it learns about the environment. **But** ...
- ... the more it explores the less it takes advantage of what it knows.

The Explore/Exploit Dilemma

- The tradeoff between doing what you (currently) think is best, and trying something new which might be better.
  - Like a menu in a foreign restaurant.
- Dilemma exists in RL, not in SL.
- Very difficult to find optimal tradeoff.

Sequential and Non-sequential Tasks

Non-sequential Tasks

- Actions do not affect inputs.
- E.g. pattern classification.

Sequential Tasks

- Actions affect which inputs will be seen in the future.
- This makes learning much more difficult.
- E.g. robot navigation.

Sequential Tasks Imply Reinforcement Learning

- SL requires a correct training set, so for sequential tasks the designer of the experiment needs to work out the long term consequences of actions.
- This means a supervised learner never faces a sequential task. The designer has already solved it and translated it into a non-sequential task.
- In RL, reward is a measure of immediate value only. The designer leaves it to the learner to learn the long term consequences of actions.
- Consequently, for sequential tasks it is often easier to invent a suitable reward function than to find a correct training set.

A Function Optimisation Problem

- Given a function f, find the value x which maximises $f(x)$.

One possible representation

- Learn to approximate the function using if-then rules:

IF	$0 \leq x < 1$	**THEN**	$f(x) = 10$
IF	$1 \leq x < 2$	**THEN**	$f(x) = 20$
$\vdots$			

- But since we only care about finding the value of x which maximises $f(x)$ (and not the other values of x) this is pointless.
- This task is suitable for, e.g., a function optimisation GA, rather than an LCS.

Summary

- LCS can be applied to a wide range of problems.
- Sequential tasks impose additional difficulties.

I Types of Problems

II Introduction to Classifier Systems

III Representation

IV Credit Assignment

V Rule Discovery

VI Conclusion

Learning Classifier Systems (LCS)

- Incremental machine learning systems suitable for both Supervised Learning and Reinforcement Learning.
- Use a population of if-then rules (called classifiers) to make decisions.
  - Where do rules come from?
- Often more than one applies at a time and conflict resolution must occur.
  - How do we choose between applicable rules?

LCS contain two learning subsystems to address these problems.

Credit Assignment Subsystem

- Evaluates the usefulness of existing rules so we can do conflict resolution.
- The utility estimate is called the *strength* of a rule.
- Typically either the Bucket Brigade or (more recently) Q-learning algorithm.

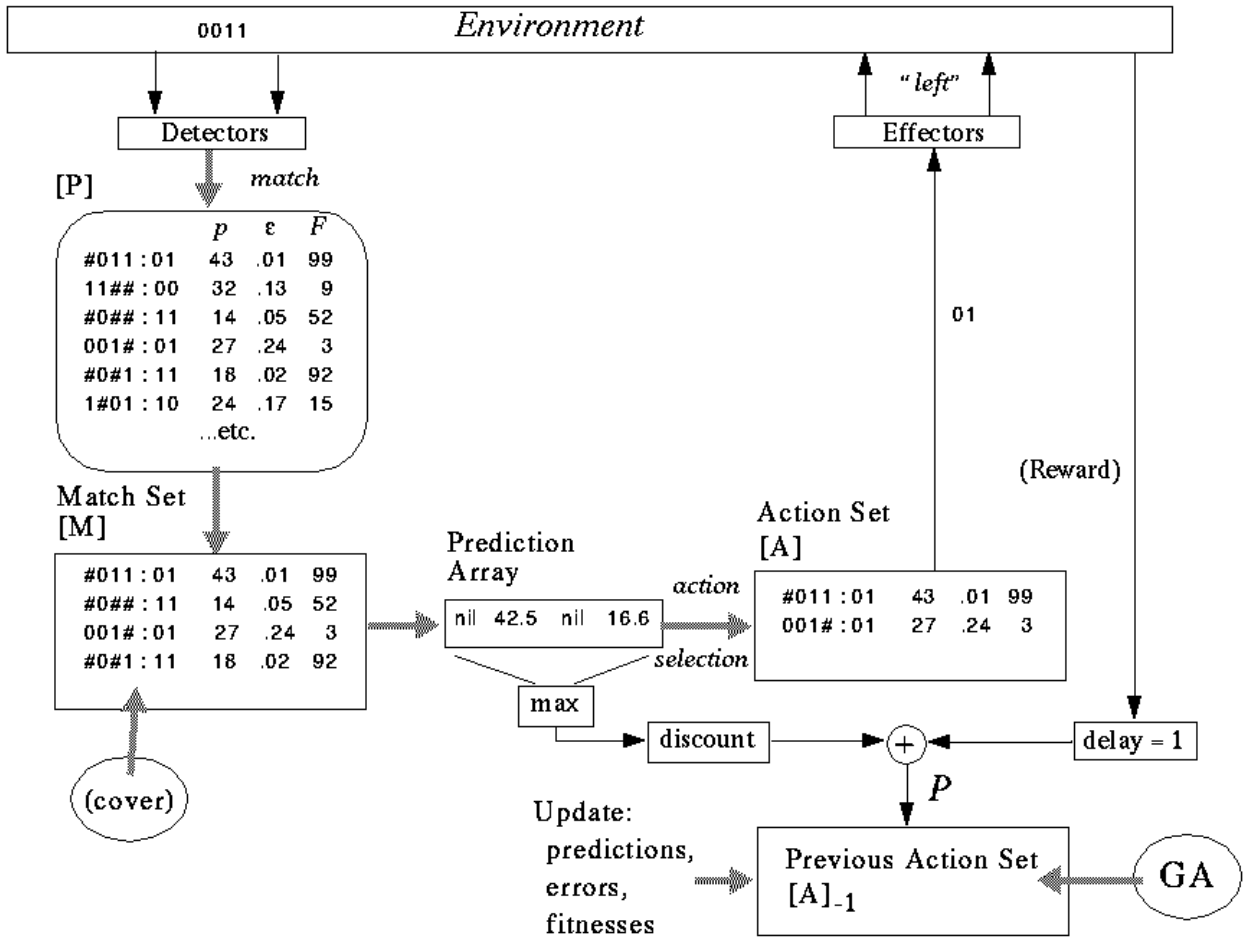
Rule Discovery Subsystem

- Generates new (hopefully useful) rules and deletes less useful ones.
- Requires a numerical estimate of the value of each rule (fitness).
- Obvious solution: use strength value generated by credit assignment system as rule's fitness.
- Typically a Genetic Algorithm.

Basic Operation of the LCS

- LCS operate in cycles:
  - LCS receives input.
  - LCS finds rules which match input (called the match set [M]).
  - The matching rules may not advocate the same action: conflict resolution occurs and an action is chosen.
    * The rules advocating the chosen action are called the action set [A].
  - LCS performs chosen action & receives feedback on it.
  - Rule utility estimates are updated based on feedback.
  - Rule discovery system (Genetic Algorithm) may be called.
- Note that rules are evaluated incrementally as inputs arrive.

Schematic Illustration of XCS



Courtesy of Stewart W. Wilson.

Types of Classifier System

There are many ways we can categorise LCS. Here are two:

- Pittsburgh vs. Michigan.
- Strength-based vs. Accuracy-based.

Pitt and Michigan LCS

Pittsburgh LCS

- A chromosome codes an entire set of rules.
- Credit assignment evaluates an entire set of rules.
- Treats RL/SL as function optimisation. **Big difference.**
  - Arguably a kind of GA not a kind of LCS.

Michigan LCS

- A chromosome codes a single rule.
- Credit assignment evaluates individual rules.

Strength-Based and Accuracy-Based LCS

Strength-Based

- Fitness = strength.
- I.e. fitter rules are those which receive higher rewards.

Accuracy-Based

- Fitness = accuracy of strength at predicting rewards.
- I.e. fitter rules are those which predict reward more accurately.

A Classifier Systems Renaissance

In 1996 Cribbs and Smith [16] claimed an LCS renaissance was underway:

- New representations.
- New credit assignment algorithms.
- New rule discovery algorithms.
- Closer connection to mainstream Reinforcement Learning (i.e. [40]).

Since then much progress has been made in all these areas.

I Types of Problems

II Introduction to Classifier Systems

III Representation

IV Credit Assignment

V Rule Discovery

VI Conclusion

Standard Ternary Language

- Each rule has one condition and one action.
- Inputs and actions are fixed length binary strings.
- Rule conditions are fixed length ternary strings from {0,1,#}.
- # is a wildcard and matches either a 1 or 0 in the input. E.g. input 010 is matched by these conditions: ###, ##0, 0##, #1#, #10, 0#0, 01#, 010.
- Conditions are sometimes called *taxa* (singular: *taxon*).

Enhanced Ternary Languages

- Negation of conditions [37, 20, 7]
  - If not(0#0) then (take action 01).
- Conjunctions of multiple conditions may be allowed in one rule [18, 20, 7]
  - If (0#0 AND 101) then (take action 01).
- Actions may contain #s which are replaced by bits in the matched input (pass-through) [17, 20, 34, 39]
  - If (0#0) then (take action 1#1).
- Partial matching [3].
- More expressive binary codings [4].

Default Hierarchies

- Code solution using general default rules and more specific exception rules [20].
- Should increase the number of solutions without increasing the size of the search space.
- Should allow gradual refinement of knowledge (by adding exception rules).

The 3-Multiplexer Function

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

Two Representations of the 3-Multiplexer

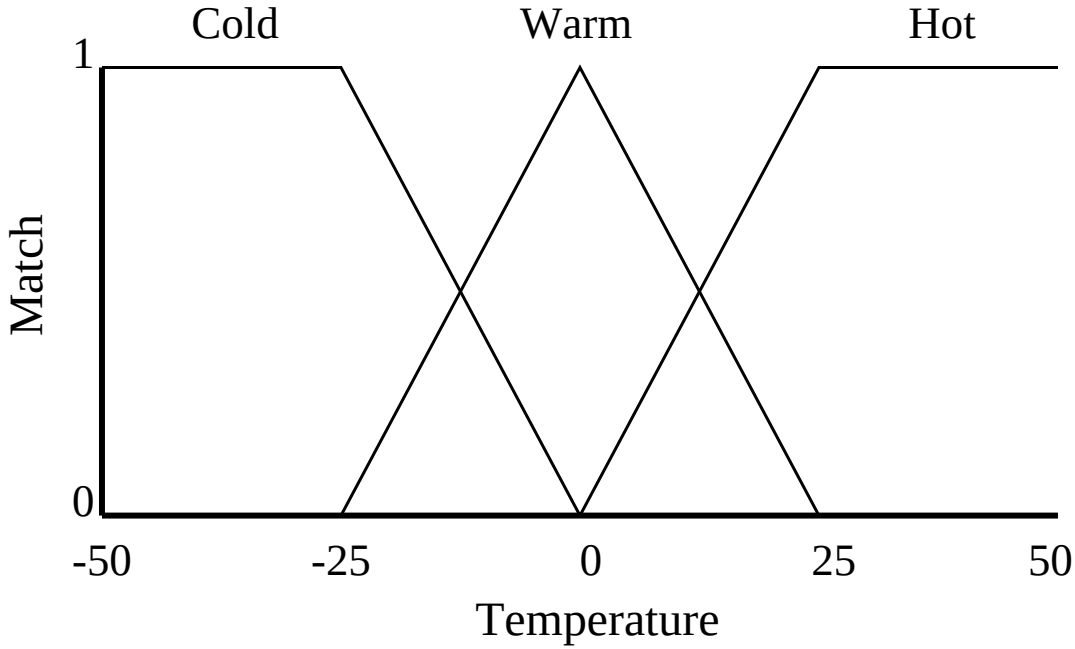
Default Hierarchy	**Best Action Map**
$0\,0\,\#\rightarrow 0$	$0\,0\,\#\rightarrow 0$
$1\,\#\,0\rightarrow 0$	$0\,1\,\#\rightarrow 1$
$\#\#\#\rightarrow 1$	$1\,\#\,0\rightarrow 0$
	$1\,\#\,1\rightarrow 1$

Default Hierarchies – Analysis

- Hard to evolve – rules need to cooperate.
- Unstable – introduce interdependence between rules.
- Complicate credit assignment, since exception rules must override defaults [44, 36].
  - Not all LCS support them (e.g. ZCS and XCS do not).
- Fewer #s doesn't mean a rule actually matches fewer states [36].
- Why must exceptions be more specific? [36]
- Not much recent interest in them.

Fuzzy Classifiers

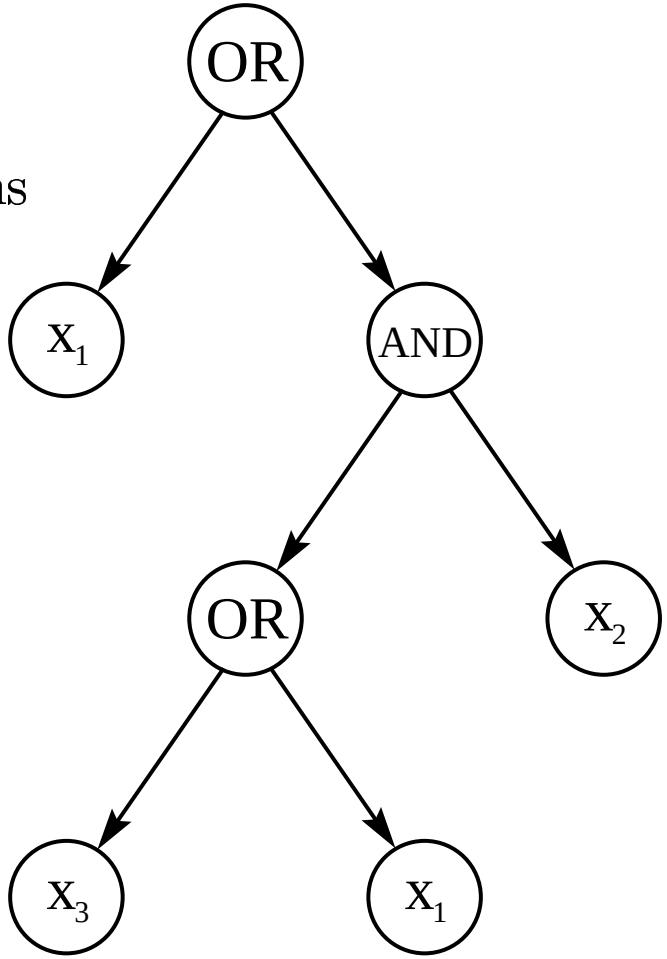
- Rules use Fuzzy Logic in matching conditions to inputs [41, 2].
- Rules match to some degree, rather than all or nothing.



Genetic Programming-Based LCS

- Use Genetic Programming to evolve conditions which compute an action [45, 30].

- E.g. evolve tree structures computing Boolean functions using {AND, OR, NOT} on inputs $\{x_1, x_2, x_3\}$.

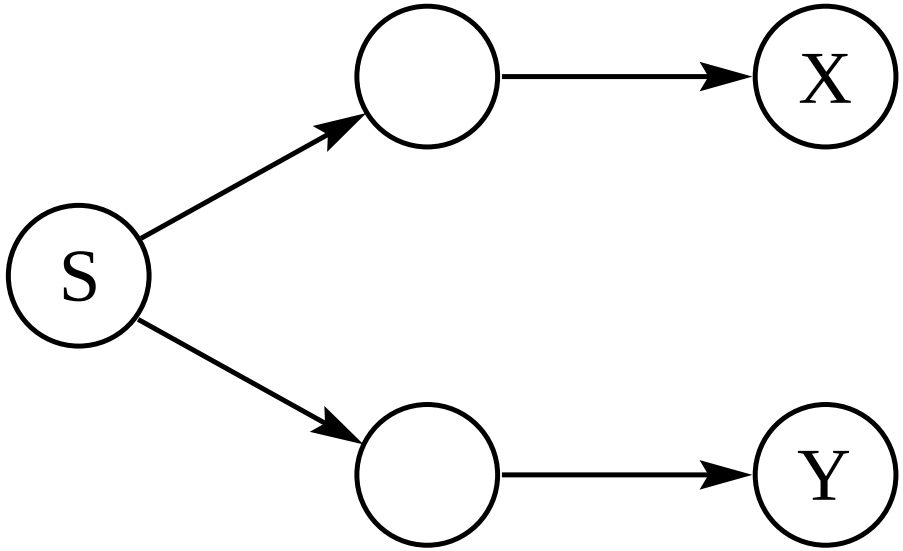


Models of the World

- A basic rule consists of a condition, action, and strength.
- In addition, a rule can contain an *expecton* – a prediction of the next state.
  - if (0 1 #) then (take action 0 1) expect (state 0 1 #).
- This allows planning and latent learning (learning in the absence of reward) e.g. [35, 19, 38, 39, 11].

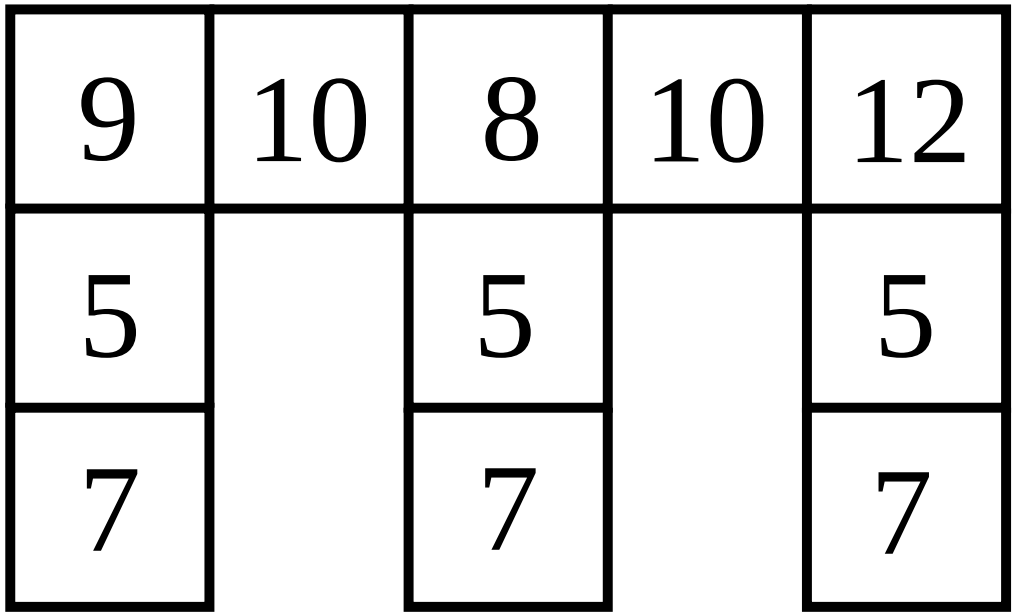
Latent Learning

- Let the learner explore the maze without reward.
- Place the learner in state X and reward it.
- Place the learner in state S and see whether it goes to X or Y.



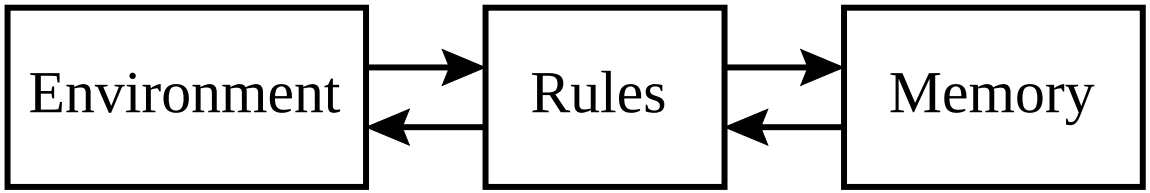
Internal Memory

- *Stimulus-response* LCS have no internal memory.
- Memory is needed to solve tasks where inputs are *perceptually aliased*, e.g. McCallum's maze [32].



Internal Memory

- Memory can be added in the form of:
  - A message list [20]
    - if (input = 001 and list contains 010) then (take action 0, post 111).
  - Bit registers [45, 29]
    - if (input = 001 and bit 1 is set) then (take action 0, clear bit 1).
- Rule actions can add messages to the list / set bits in the register.
- Rule conditions can match messages / register settings.



Macroclassifiers

- Use **one** rule to represent many identical virtual rules [46].
- This reduces runtime and provides interesting statistics.
- Empirically, macroclassifiers perform essentially as the equivalent 'micro'classifiers do [23].

Example of Macroclassifiers

Without Macroclassifiers

Rule	Cond.	Action	Strength
m	##0011	1	200.0
m′	##0011	1	220.0
n	##0011	0	100.0
o	001110	1	100.0

With Macroclassifiers

Rule	Cond.	Action	Strength	Numerosity
m	##0011	1	200.0	2
n	##0011	0	100.0	1
o	001110	1	100.0	1

I Types of Problems

II Introduction to Classifier Systems

III Representation

IV Credit Assignment

V Rule Discovery

VI Conclusion

ZCS

Traditional LCS have had mixed success, and are complex and difficult to analyse. Wilson created a minimalist LCS called ZCS [45].

- No internal message list (a "stimulus-response" LCS).
- Simple condition syntax: 1 condition per rule. No negation.
- No pass-through in actions.
- Action selection and rule updates simplified.

Wilson demonstrated that, despite its minimalism, ZCS was able to learn two tasks, albeit suboptimally [45].

Problems with ZCS

Overgeneral rules.

- Rules which can be improved by replacing #s with $0, 1$.
- How can we distinguish between an overgeneral "guesser" which gets x reward on average, and a consistently optimal rule which gets x because it matches inherently low-rewarding inputs?

Greedy classifier creation.

- More rules are allocated to states with higher rewards. Low-rewarding states may have few or no rules.†

†LCS need to know how to act in each state they encounter – this is like massively multimodal function optimisation.

Strong Overgeneral Rules

Interaction: *Strong overgeneral* rules [15, 27].

- An overgeneral with greater strength than some not-overgeneral rule with which it competes.
- Sometimes acts incorrectly, but has greater influence in action selection than its correct competitor.
- Moral: What's best on average isn't necessarily best right now.

Fit Overgeneral Rules

If strength is used as fitness, strong overgenerals are *fit overgenerals*:

- An overgeneral with greater fitness than some not-overgeneral rule with which it competes.
- Sometimes acts incorrectly, but has greater influence in reproduction that its correct competitor.

Strong and fit overgenerals are a major problem for strength-based LCS [15, 27].

When will ZCS work?

ZCS's fitness sharing can reduce problems with fit overgenerals [9]. Nonetheless, my current evaluation is:

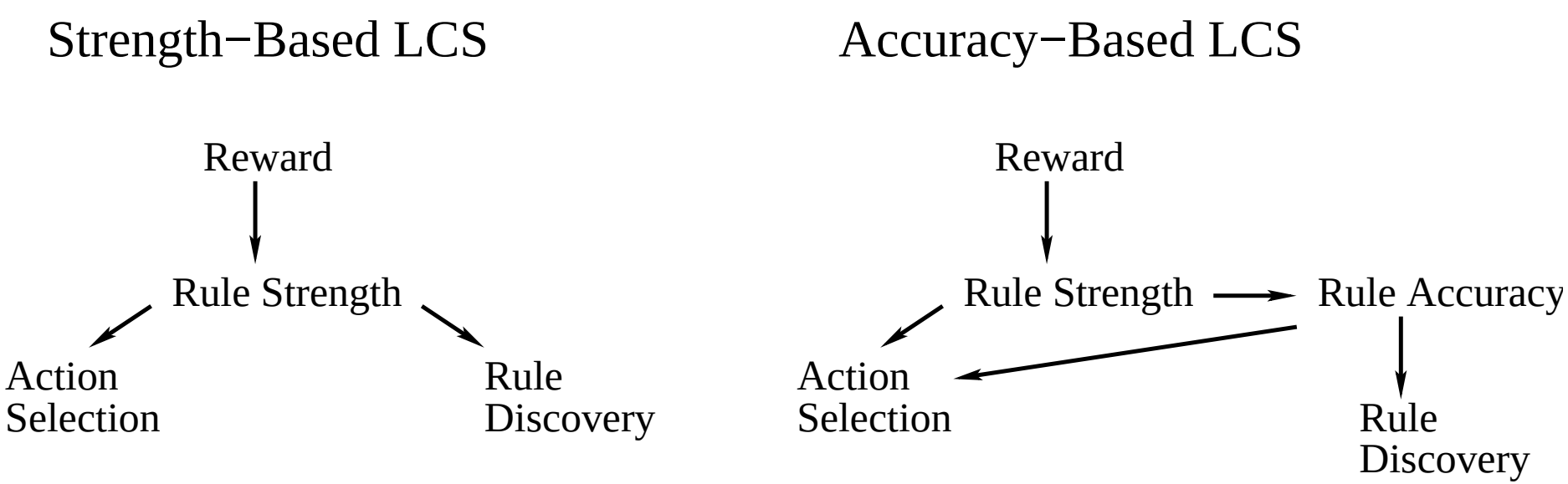
- ZCS should work on many non-sequential tasks [27].
- ZCS will not work well on some sequential tasks [27].

The solution: use accuracy-based fitness. Wilson created an accuracy-based LCS called XCS.

XCS: Accuracy-Based Fitness

- Wilson's XCS [46] is based on his earlier ZCS.
  - Action selection is still done on the basis of strength.
  - GA fitness is now based on how accurately a rule's strength predicts the rewards it receives.

Two Approaches to Fitness Calculation (cont'd)



XCS's Accuracy-Based Fitness

1. Strength (prediction): $s_j \leftarrow s_j + \beta(R - s_j)$

where $0 < \beta \leq 1$ controls the learning rate, R is the reward from the environment, and the update is for non-sequential tasks.

---

2. Prediction error: $\varepsilon_j \leftarrow \varepsilon_j + \beta(|R - s_j| - \varepsilon_j)$

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

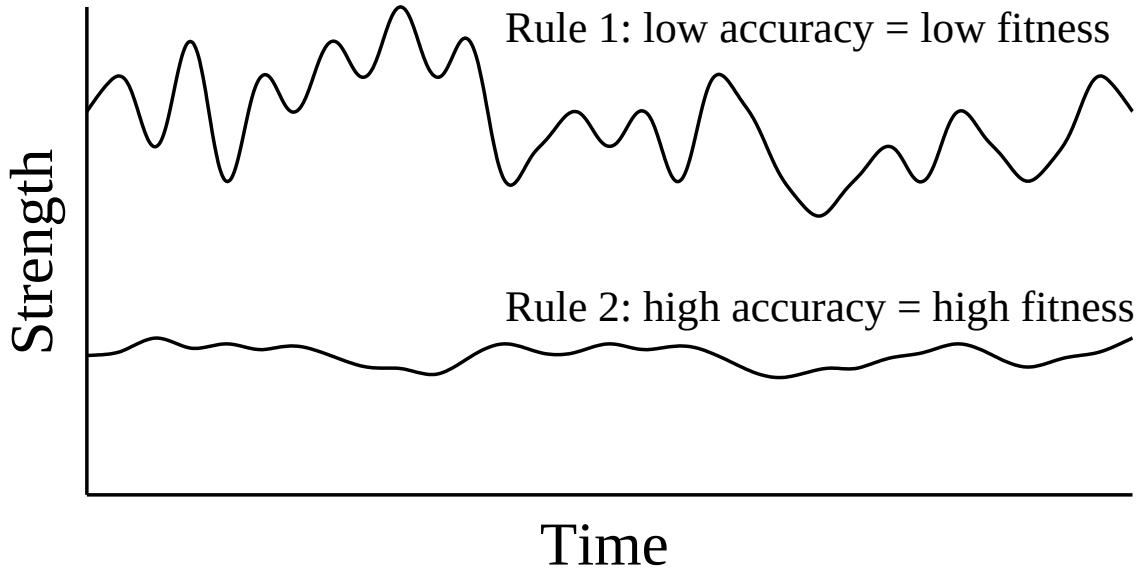
where ε_o controls the tolerance for prediction error and $0 < \alpha < 1$ and $0 < v$ control the rate of decline in accuracy when ε_o is exceeded.

4. Relative accuracy: $\kappa'_j = \kappa_j \; / \sum_{x \in [A]} \kappa_x$

5. Fitness: $F_j \leftarrow F_j + \beta(\kappa'_j - F_j)$

Accuracy as Consistency

- We can think of strength as prediction of reward and accuracy as consistency of a rule's strength over time.



- With strength-based fitness the magnitude of reward influences fitness, but with accuracy-based fitness it does not.

Main Consequence of Accuracy-Based Fitness

- Overgeneral rules are updated towards different rewards, so their strength is never a good predictor of the rewards they receive.
- This means they have low accuracy, and so low fitness. Low fitness means they don't reproduce much, and tend to die out.
- So ... *in XCS overgeneral rules have low fitness and so tend to die out.*

Other Consequences of Accuracy-Based Fitness

- The fitness of a rule does not depend on the magnitude of its strength. Consequently, XCS forms *complete maps* of the input/action space: all actions in all states are advocated by some rule.
  - ...so greedy classifier creation isn't a problem.
  - ...complete maps may help with Temporal Difference learning and exploration control in Reinforcement Learning [27].
- No default hierarchies are possible since all overgeneral rules are unfit.
  - ...but D.H.s are problematic anyway (unstable and hard to evolve).

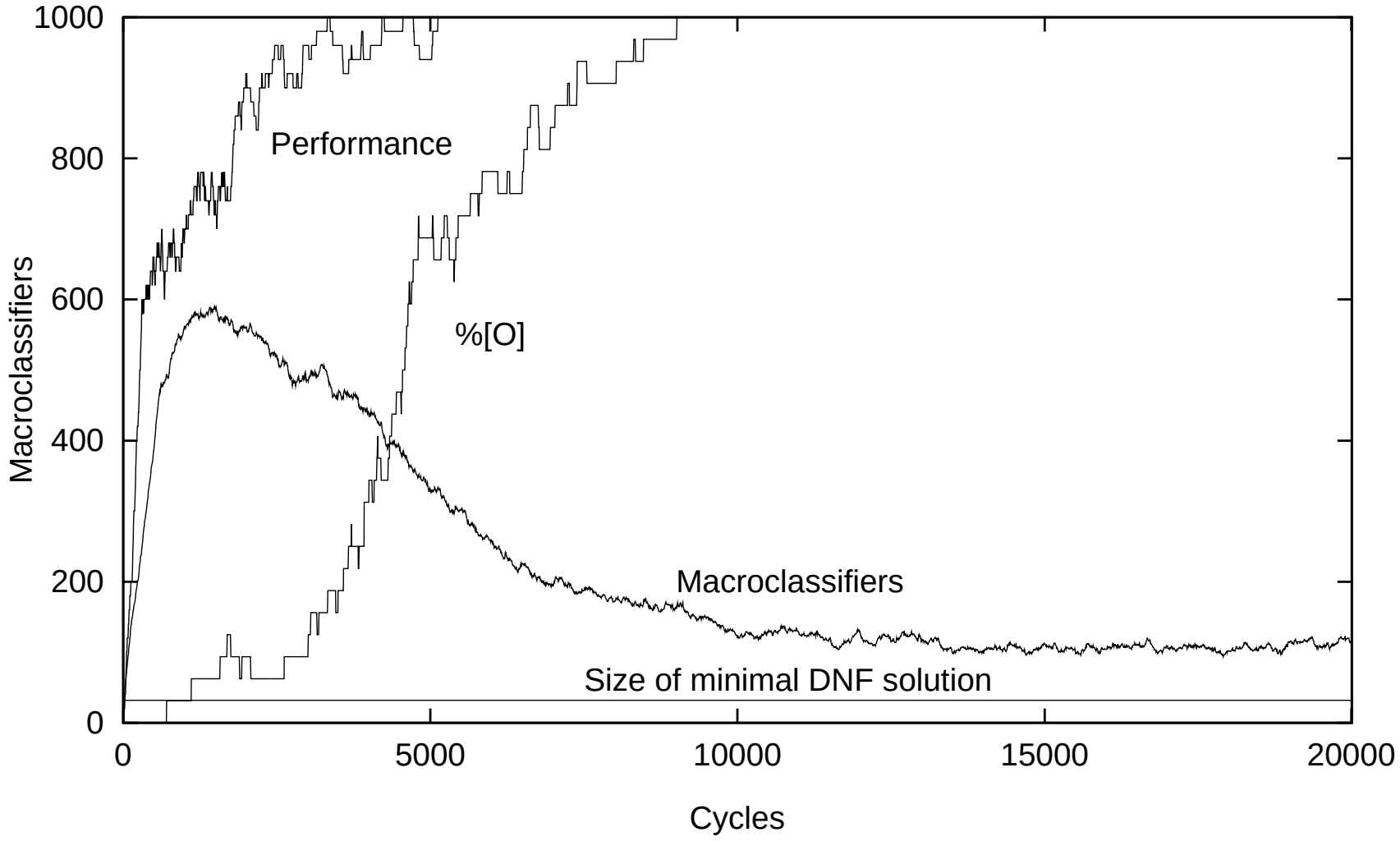
Three Representations of the 3-Multiplexer

Default Hierarchy	Best Action Map	Complete Map
0 0 # → 0	0 0 # → 0	0 0 # → 0
1 # 0 → 0	0 1 # → 1	0 0 # → 1
# # # → 1	1 # 0 → 0	0 1 # → 0
	1 # 1 → 1	0 1 # → 1
		1 # 0 → 0
		1 # 0 → 1
		1 # 1 → 0
		1 # 1 → 1

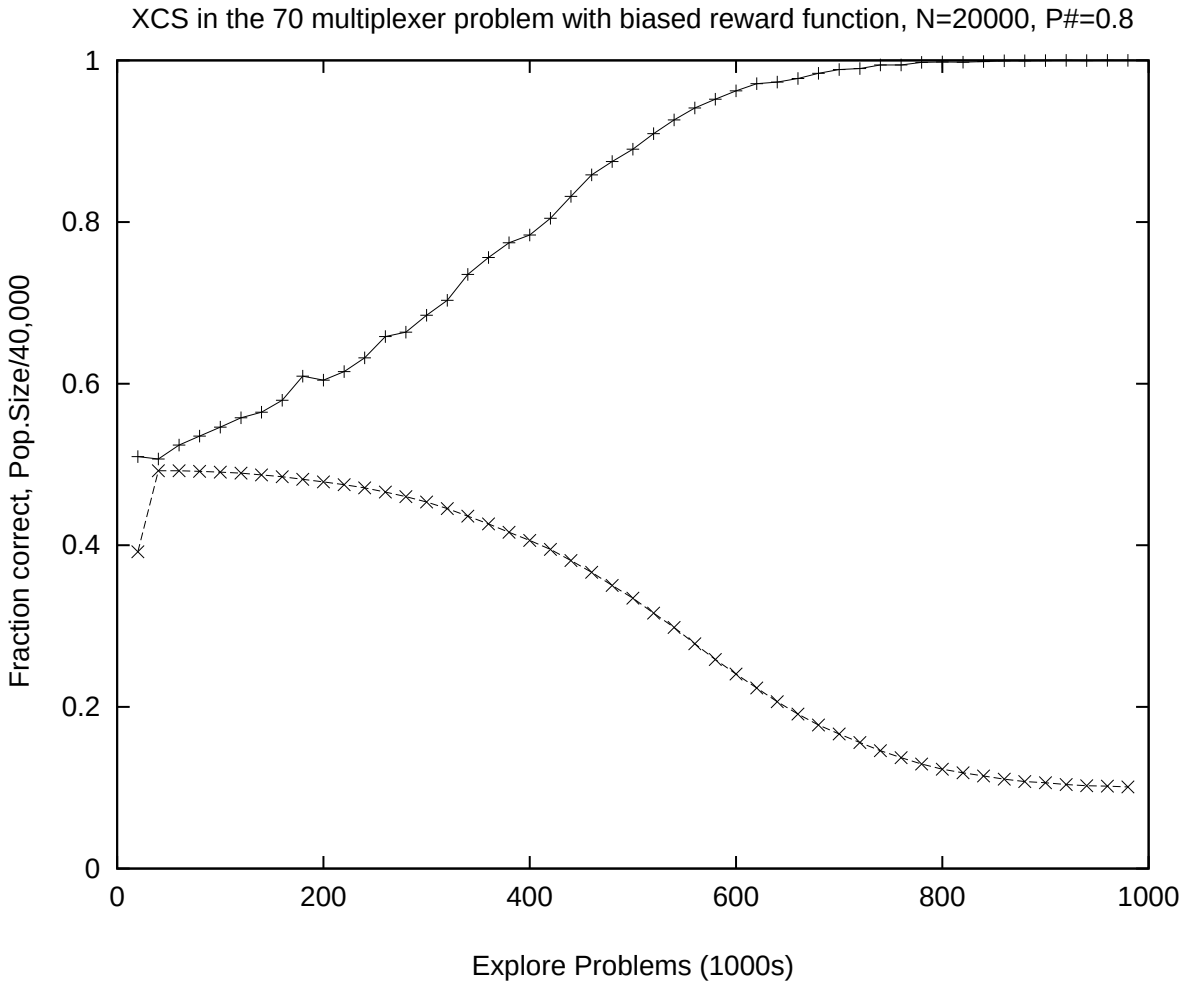
How XCS Works

- XCS has a bias towards accurate rules (thanks to fitness definition).
  - Hence, no strong or fit overgenerals.
- XCS has a bias towards general rules (thanks to niche GA).
- Consequently, XCS tends to find accurate, general rules.
  - This makes it easier to "see the knowledge" [47].
  - Accurate generalisation is essential to learning systems [47].
- XCS also has a strong implicit bias against overlapping rules (rules which match the same inputs) [24].

XCS on the 11 Multiplexer



XCS on the 70 Multiplexer



A search space of $3^{70} \cdot 2$ rules! [14].

XCS as an Evolutionary System

In XCS we see examples of:

- **Fitness sharing** Fitness is explicitly shared between rules in an action set (in the relative accuracy update).
- **Co-evolution** The fitness of a rule depends heavily on the others in the population because of fitness sharing and the bias against overlapping rules.
- **Crowding** A rule's deletion probability is proportional to its estimate of the average size of the action sets it occurs in. So rules are more likely to be deleted from parts of the space where there are many rules, and less likely to be deleted from parts where there are few rules.

Booker's Endogenous Fitness LCS

A novel approach to calculating rule fitness [5, 6]:

- Rather than explicitly estimating how good a rule is (i.e. estimating strength or fitness), just let it collect resources.
- Let it spend its resources on reproduction.
- The best rules will accumulate the most resources and reproduce the most.
- Initial results suggest it performs similarly to XCS.

I Types of Problems

II Introduction to Classifier Systems

III Representation

IV Credit Assignment

V Rule Discovery

VI Conclusion

Steady State Genetic Algorithms

- Generational GAs replace the whole population from one generation to the next.
- Steady State GAs (SSGAs) replace part of the population...
  - ... but the part can be large (e.g. 80%), so the difference can be minor.
- LCS typically use a SSGA to minimise disruption of the population.
  - Needed because rules are evaluated incrementally.
- A SSGA applies selective pressure at two points:
  - For reproduction.
  - For deletion.

Niche Genetic Algorithm

- In a *panmictic* GA any rule can be selected for reproduction.
- In a niche GA, mating is restricted to rules in the action set.
- The idea is that mating related rules should be more effective.
- Other effects:
  - Introduces a strong bias towards more general rules, since they match more inputs.
  - Introduces pressure against overlapping rules, since they compete.
  - Helps maintain diversity, since competition occurs within subsets of population.

Subsumption Deletion

- A rule x *logically subsumes* a rule y when x matches a superset of the inputs y matches, and they have the same action.
  - E.g., 00#→0 subsumes 000→0 and 001→0.
- In XCS a rule is allowed to subsume another if:
  - It logically subsumes it.
  - It is accurate (has low prediction error).
  - It is experienced (has been evaluated sufficiently) – so we can be sure it is accurate.

Two Checks for Subsumption

GA Subsumption

- When a new rule is generated, check to see if its parents subsume it.
- This constrains accurate parents to only produce more general children.

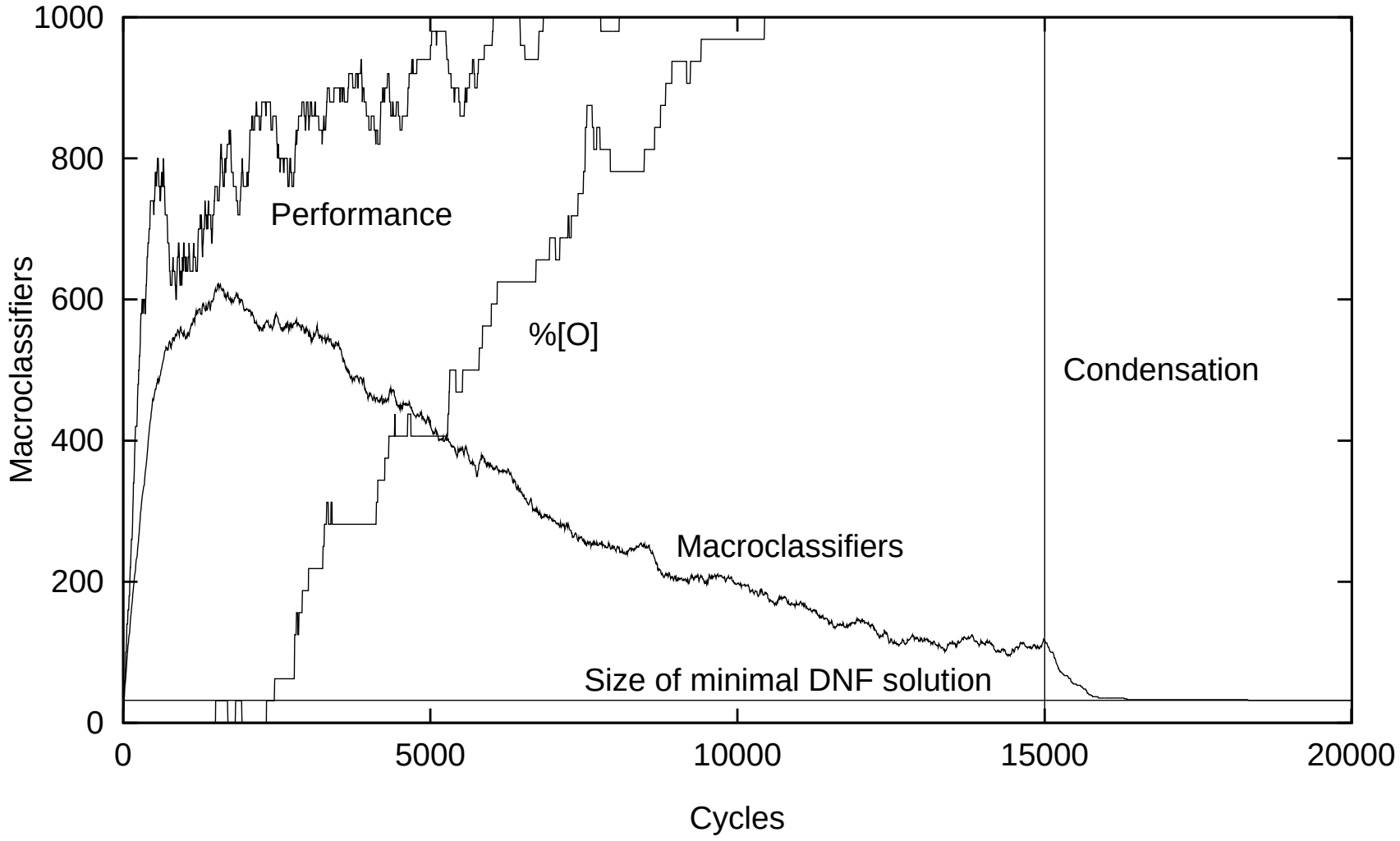
Action Set Subsumption

- Check rules in each action set, and let most general of the accurate, experienced rule subsume other rules.
- This removes redundant specific rules from the population.

Condensation

- An evolved population normally contains many redundant and (transient) low-fitness rules.
- We can remove them using condensation:
  - run the system with crossover and mutation turned off [46, 23].
  - I.e. only clone and delete exsting rules.

The Effect of Condensation – XCS on the 11 Multiplexer



Non-Evolutionary LCS

- Although LCS were originally conceived as a way of applying a GA to a problem [21], not all LCS include a GA!
- E.g. ACS, which uses heuristics to create and refine rules [38, 11].
  - Later supplemented by a GA [12, 13].

I Types of Problems

II Introduction to Classifier Systems

III Representation

IV Credit Assignment

V Rule Discovery

VI Conclusion

Three Types of LCS Compared

Pittsburgh LCS

- Can solve complex problems [43].
- Are slow (since each chromosome is a set of rules).

Strength-Based Michigan LCS

- Can have difficulties with complex problems (e.g. [43, 15, 27]).
- Are fast (since each chromosome is an individual rule).

Accuracy-Based Michigan LCS (i.e. XCS)

- Can solve complex problems [27].
- Are fast (since each chromosome is an individual rule).

Conclusion

- There's a classifier systems renaissance underway.
- The LCS is a very broad paradigm – not just a GA-based system.
  - Many representations are possible.
  - Many credit assignment schemes are possible.
  - Many rule discovery systems are possible.
- Links to Reinforcement Learning and Machine Learning are important, and growing.
- XCS is an active research area.

LCS Resources

- The LCS Web [1].
- The LCS Mailing List [22].
- The LCS Bibliography [26] and Archive [28].
- See [25, 10, 33, 8] and the proceedings of GECCO and the International Workshop on Learning Classifier Systems (IWLCS) for more.

References

[1] Alwyn Barry. The Learning Classifier Systems Web. University of Bath, 2004. http://www.cs.bath.ac.uk/~amb/LCSWEB/.

[2] Andrea Bonarini. An Introduction to Learning Fuzzy Classifier Systems. In Lanzi et al. [31], pages 83–104.

[3] Lashon B. Booker. Improving the performance of genetic algorithms in classifier systems. In John J. Grefenstette, editor, *Proceedings of the 1st International Conference on Genetic Algorithms and their Applications (ICGA-85)*, pages 80–92, Pittsburgh, PA, July 1985. Lawrence Erlbaum Associates.

[4] Lashon B. Booker. Representing attribute-based concepts in a classifier system. In Gregory J. E. Rawlins, editor, *Proceedings of the First Workshop on Foundations of Genetic Algorithms*, pages 115–127, San Mateo, July 15–18 1991. Morgan Kaufmann.

[5] Lashon B. Booker. Do We Really Need to Estimate Rule Utilities in Classifier Systems? In Lanzi et al. [31], pages 125–142.

[6] Lashon B. Booker. Classifier systems, endogenous fitness, and delayed rewards: A preliminary investigation. In Lee Spector, Erik D. Goodman, Annie Wu, W.B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 921–926, San Francisco, California, USA, 7-11 July 2001.

[7] Lashon B. Booker, David E. Goldberg, and John H. Holland. Classifier systems and genetic algorithms. *Artificial Intelligence*, 40:235–282, 1989.

[8] Larry Bull, editor. *Applications of Learning Classifier Systems*. Springer, 2004.

[9] Larry Bull and Jacob Hurst. ZCS redux. *Evolutionary Computation*, 10(2):185–205, 2002.

[10] Larry Bull, Pier Luca Lanzi, and Wolfgang Stolzmann (guest editors). Journal of soft computing, special issue on learning classifier systems, 6(3–4), 2002.

[11] Martin V. Butz. *Anticipatory learning classifier systems*. Kluwer Academic Publishers, 2002.

[12] Martin V. Butz, David E. Goldberg, and Wolfgang Stolzmann. Introducing a Genetic Generalization Pressure to the Anticipatory Classifier System – Part 1: Theoretical Approach. In Whitely et al. [42], pages 34–41. Also Technical Report 2000005 of the Illinois Genetic Algorithms Laboratory.

[13] Martin V. Butz, David E. Goldberg, and Wolfgang Stolzmann. Introducing a Genetic Generalization Pressure to the Anticipatory Classifier System – Part 2: Performance Analysis. In Whitely et al. [42], pages 42–49. Also Technical Report 2000006 of the Illinois Genetic Algorithms Laboratory.

[14] Martin V. Butz, Tim Kovacs, Pier Luca Lanzi, and Stewart W. Wilson. How XCS Evolves Accurate Classifiers. In Lee Spector, Erik D. Goodman, Annie Wu, W. B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H Garzon, and Edmund Burke, editors, *GECCO-2001: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 927–934. Morgan Kaufmann, 2001.

[15] Dave Cliff and Susi Ross. Adding Temporary Memory to ZCS. *Adaptive Behavior*, 3(2):101–150, 1995.

[16] Henry Brown Cribbs III and Robert E. Smith. Classifier system renaissance: New analogies, new directions. In John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 547–552, Stanford University, CA, USA, 28–31 July 1996. MIT Press.

[17] Stephanie Forrest. *A study of parallelism in the classifier system and its application to classification in KL-ONE semantic networks.* PhD thesis, University of Michigan, Ann Arbor, MI, 1985.

[18] David E. Goldberg. *Computer-Aided Gas Pipeline Operation using Genetic Algorithms and Rule Learning.* PhD thesis, The University of Michigan, 1983.

[19] John H. Holland. Concerning the emergence of tag-mediated lookahead in classifier systems. *Physica D*, 42:188–201, 1990.

[20] John H. Holland, Keith J. Holyoak, Richard E. Nisbett, and Paul R. Thagard. *Induction. Processes of Inference, Learning and Discovery.* The MIT Press, 1986.

[21] John H. Holland and J. S. Reitman. Cognitive systems based on adaptive algorithms. In D. A. Waterman and F. Hayes-Roth, editors, *Pattern-directed Inference Systems.* New York: Academic Press, 1978. Reprinted in: Evolutionary Computation. The Fossil Record. David B. Fogel (Ed.) IEEE Press, 1998. ISBN: 0-7803-3481-7.

[22] John Holmes. The Learning Classifier Systems Mailing List. The University of Pennsylvania, 2004. cslist@cceb.med.upenn.edu. Mail John Holmes <jholmes@cceb.med.upenn.edu> to subscribe.

[23] Tim Kovacs. Evolving Optimal Populations with XCS Classifier Systems. Master's thesis, University of Birmingham, Birmingham, UK, 1996.

[24] Tim Kovacs. What should a classifier system learn? In *Proceedings of the 2001 Congress on Evolutionary Computation*, volume 2, pages 775–782. IEEE Press, 2001.

[25] Tim Kovacs. Learning Classifier Systems Resources. *Journal of Soft Computing*, 6(3–4):240–243, 2002.

[26] Tim Kovacs. A Learning Classifier Systems Bibliography. Department of Computer Science, University of Bristol, 2004. http://www.cs.bris.ac.uk/~kovacs/lcs/search.html.

[27] Tim Kovacs. *Strength or Accuracy: Credit Assignment in Learning Classifier Systems*. Springer, 2004.

[28] Tim Kovacs. The Electronic Learning Classifier Systems Archive. Department of Computer Science, University of Bristol, 2004. http://www.cs.bris.ac.uk/~kovacs/lcs/lcs.archive.html.

[29] Pier Luca Lanzi. An analysis of the memory mechanism of XCSM. In John R. Koza, Wolfgang Banzhaf, Kumar Chellapilla, Kalyanmoy Deb, Marco Dorigo,

David B. Fogel, Max H. Garzon, David E. Goldberg, Hitoshi Iba, and Rick Riolo, editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 643–651. Morgan Kaufmann, 1998. http://ftp.elet.polimi.it/people/lanzi/gp98.ps.gz.

[30] Pier Luca Lanzi. Extending the Representation of Classifier Conditions Part II: From Messy Coding to S-Expressions. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 345–352. Morgan Kaufmann, 1999.

[31] Pier Luca Lanzi, Wolfgang Stolzmann, and Stewart W. Wilson, editors. *Learning Classifier Systems. From Foundations to Applications*, volume 1813 of *LNAI*. Springer-Verlag, Berlin, 2000.

[32] Andrew McCallum. *Reinforcement Learning with Selective Perception and Hidden State*. PhD thesis, University of Rochester, 1996.

[33] Wolfgang Stolzmann Pier Luca Lanzi and Stewart W. Wilson (guest editors). Journal of evolutionary computing, special issue on learning classifier systems, 11(3), 2003.

[34] Rick L. Riolo. CFS-C: A Package of Domain-Independent Subroutines for Implementing Classifier Systems in Arbitrary User-Defined Environments. Technical report, University of Michigan, 1988.

[35] Rick L. Riolo. Lookahead planning and latent learning in a classifier system. In *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior (SAB-90)*, pages 316–326. The MIT Press, 1991.

[36] Robert E. Smith and David E. Goldberg. Variable default hierarchy separation in a classifier system. In Gregory J. E. Rawlins, editor, *Proceedings of the First Workshop on Foundations of Genetic Algorithms*, pages 148–170, San Mateo, July 15–18 1991. Morgan Kaufmann.

[37] S. F. Smith. *A Learning System Based on Genetic Adaptive Algorithms*. PhD thesis, University of Pittsburgh, 1980.

[38] Wolfgang Stolzmann. Learning classifier systems using the cognitive mechanism of anticipatory behavioral control, detailed version. In *Proceedings of the First European Workshop on Cognitive Modelling*, pages 82–89. Berlin: TU, 1996.

[39] Wolfgang Stolzmann. An Introduction to Anticipatory Classifier Systems. In Lanzi et al. [31], pages 175–194.

[40] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[41] Manuel Valenzuela-Rendón. The Fuzzy Classifier System: a Classifier System for Continuously Varying Variables. In Lashon B. Booker and Richard K. Belew, editors, *Proceedings of the 4th International Conference on Genetic Algorithms (ICGA91)*, pages 346–353. Morgan Kaufmann, July 1991.

[42] Darrell Whitely, David Goldberg, Erick Cantú-Paz, Lee Spector, Ian Parmee, and Hans-Georg Beyer, editors. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*. Morgan Kaufmann, 2000.

[43] Jason R. Wilcox. Organizational Learning within a Learning Classifier System. Master's thesis, University of Illinois, 1995. Also Technical Report No. 95003 IlliGAL.

[44] Stewart W. Wilson. Bid competition and specificity reconsidered. *Complex Systems*, 2:705–723, 1989.

[45] Stewart W. Wilson. ZCS: A Zeroth Level Classifier System. *Evolutionary Computation*, 2(1):1–18, 1994.

[46] Stewart W. Wilson. Classifier Fitness Based on Accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.

[47] Stewart W. Wilson. Generalization in the XCS classifier system. In John Koza et al., editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 665–674. Morgan Kaufmann, 1998.