Structure and Function of the XCS Classifier System

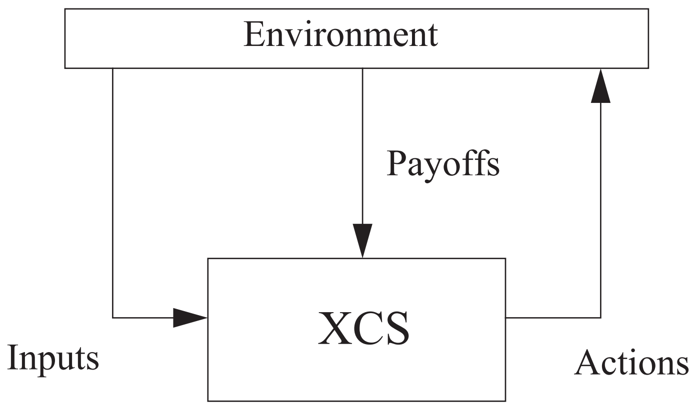
Stewart W. Wilson
Prediction Dynamics
Concord, MA
wilson@prediction-dynamics.com

What is it?

- Learning machine (program).
- Minimum *a priori*.
- “On-line”.
- Capture regularities in environment.

What does it learn?

To get reinforcements (“rewards”, “payoffs”)



(Not “supervised” learning—no prescriptive teacher.)

What inputs and outputs?

Inputs:

Now binary, e.g., 100101110

—like thresholded sensor values.

Later continuous, e.g., <43.0 92.1 7.4 ... 0.32>

Outputs:

Now discrete decisions or actions,

e.g., 1 or 0 (“yes” or “no”),

“forward”, “back”, “left”, “right”

Later continuous, e.g., “head 34 degrees left”

What’s going on inside?

XCS contains rules (called *classifiers*), some of which will match the current input. An action is chosen based on the predicted payoffs of the matching rules.

<condition>:<action> => <prediction>.

Example: 01#1## : 1 => 943.2

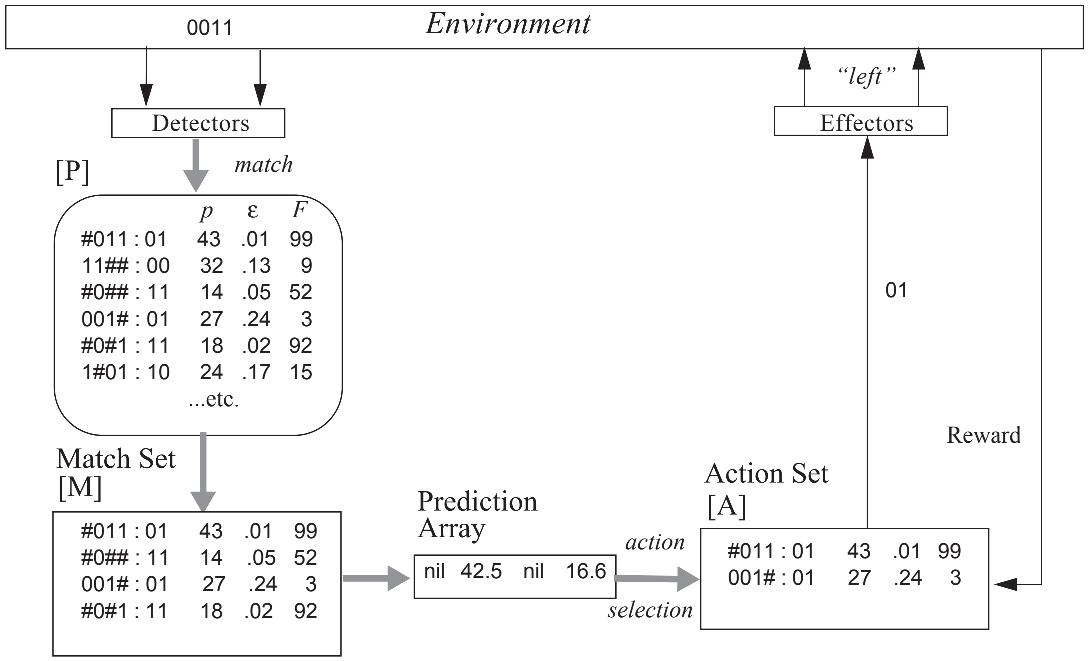
Note this rule matches more than one input string:

010100
010110
010101
011111
011100
011101
011110
011111.

This adaptive “rule-based” system contrasts with “PDP” systems such as NNs in which knowledge is distributed.

XCS

How does the performance cycle work?



- For each action in [M], classifier predictions *p* are weighted by fitnesses *F* to get system's net prediction in the prediction array.
- Based on the system predictions, an action is chosen and sent to the environment.
- Some reward value is returned.

How do rules acquire their predictions?

1. By "updating" the current estimate.

   For each classifier C_j in the current [A],

   $$p_j \leftarrow p_j + \alpha(R - p_j),$$

   where R is the current reward and α is the learning rate.

   This results in p_j being a "recency weighted" average of previous reward values:

   $$p_j(t) = \alpha R(t) + \alpha(1-\alpha)R(t-1) + \alpha(1-\alpha)^2 R(t-2) + \ldots + (1-\alpha)^t p_j(0).$$

2. And by trying different actions, according to an *explore/exploit* regime.

   A typical regime chooses a random action with probability 0.5.

   Exploration (e.g., random choice) is necessary in order to learn anything. But exploitation—picking the highest-prediction action is necessary in order to make best use of what is learned.

   There are many possible explore/exploit regimes, including gradual changeover from mostly explore to mostly exploit.

Where do the rules come from?

- Usually, the “population” [P] is initially empty. (It can also have random rules, or be seeded.)
- The first few rules come from “covering”: if no existing rule matches the input, a rule is created to match, something like imprinting.

  Input: 11000101

  Created rule: 1##0010# : 3 => 10
  Random #’s and action, low initial prediction.

- But primarily, new rules are derived from existing rules.

How are new rules derived?

- Besides its prediction p_j, each classifier's *error* and *fitness* are regularly updated.

  Error: $\varepsilon_j \leftarrow \varepsilon_j + \alpha(|R - p_j| - \varepsilon_j)$.

  Accuracy: $\kappa_j \equiv \varepsilon_j^{-n}$ if $\varepsilon_j > \varepsilon_0$, otherwise ε_0^{-n}

  Relative accuracy: $\kappa_j' \equiv \kappa_j / \left(\sum_i \kappa_i\right)$, over [A].

  Fitness: $F_j \leftarrow F_j + \alpha(\kappa_j' - F_j)$.

- Periodically, a *genetic algorithm* (GA) takes place in [A].

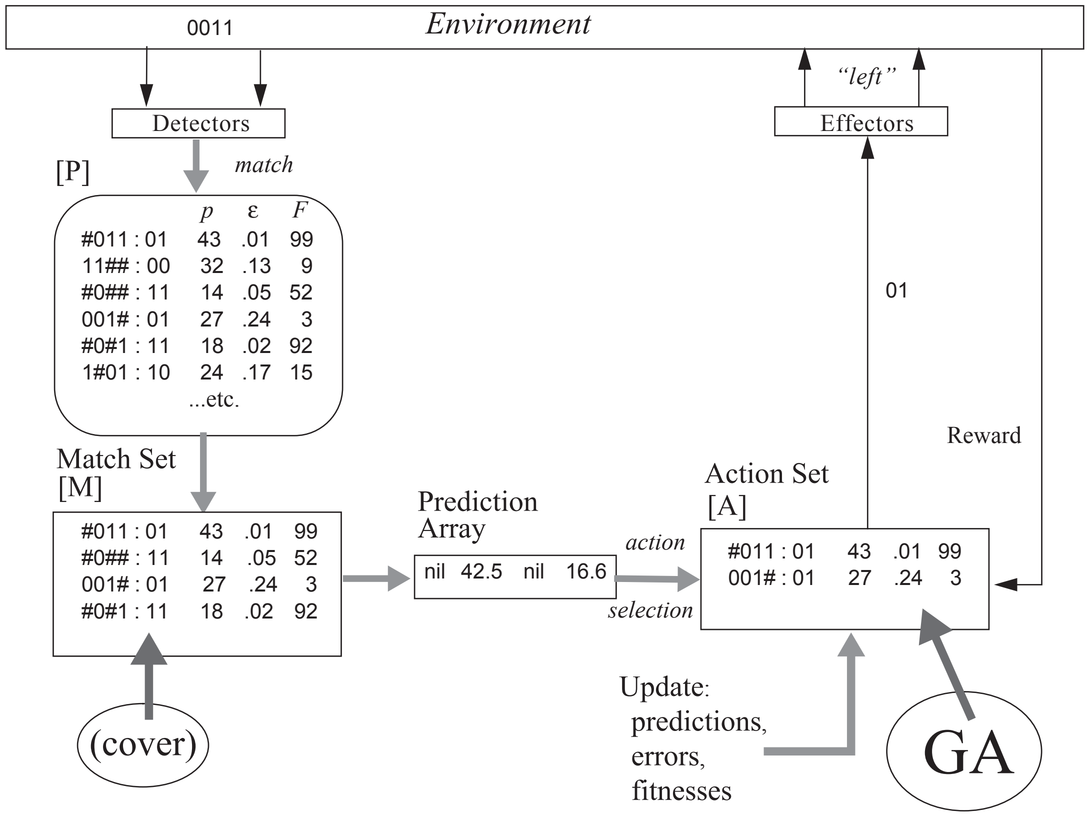
  Two classifiers C_i and C_j are selected with probability proportional to fitness. They are copied to form C_i' and C_j'.

  With probability χ, C_i' and C_j' are *crossed* to form C_i'' and C_j'', e.g.,

  $$\begin{array}{c|c} 1\ 0\ \#\ \# & 1\ 1 : 1 \\ \#\ 0\ 0\ 0 & 1\ \# : 1 \end{array} \quad \Rightarrow \quad \begin{array}{c} 1\ 0\ \#\ \#\ 1\ \# : 1 \\ \#\ 0\ 0\ 0\ 1\ 1 : 1 \end{array}$$

  C_i'' and C_j'' (or C_i' and C_j' if no crossover occurred), possibly mutated, are added to [P].

Can I see the overall process?



What happens to the “parents”?

They remain in [P], in competition with their offspring.

But two classifiers are *deleted* from [P] in order to maintain a constant population size.

Deletion is probabilistic, with probability proportional to, e.g.:

- A classifier’s average action set size a_j—estimated and updated like the other classifier statistics.
- a_j/F_j, if the classifier has been updated enough times, otherwise a_j/F_{ave}, where F_{ave} is the mean fitness in [P].

—And other arrangements, all with the aim of balancing resources (classifiers) devoted to each niche ([A]), but also eliminating low fitness classifiers rapidly.

What are the results like? — 1

Basic example for illustration: Boolean 6-multiplexer.

1 0 1 0 0 1 $\rightarrow$ $\boxed{F_6}$ $\rightarrow$ 0

1 0 1 0 0 1

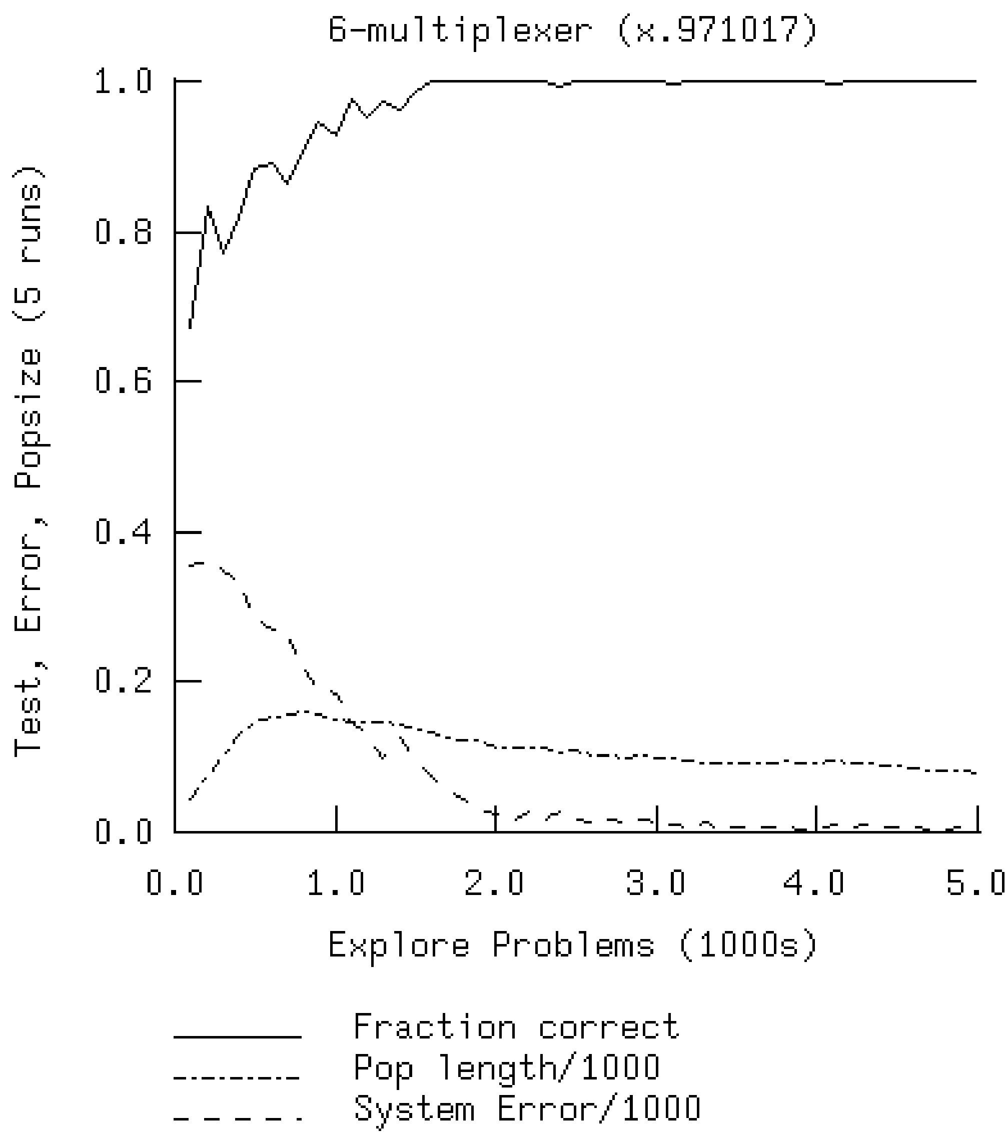
$$F_6 = x_0'x_1'x_2 + x_0'x_1x_3 + x_0x_1'x_4 + x_0x_1x_5$$

$\boldsymbol{l = k + 2^k}$ $\quad \boldsymbol{k} > 0$

$$\begin{aligned} F_{20} = {} & x_0'x_1'x_2'x_3'x_4 + x_0'x_1'x_2'x_3x_5 + \\ & x_0'x_1'x_2x_3'x_6 + x_0'x_1'x_2x_3x_7 + \\ & x_0'x_1x_2'x_3'x_8 + x_0'x_1x_2'x_3x_9 + \\ & x_0'x_1x_2x_3'x_{10} + x_0'x_1x_2x_3x_{11} + \\ & x_0x_1'x_2'x_3'x_{12} + x_0x_1'x_2'x_3x_{13} + \\ & x_0x_1'x_2x_3'x_{14} + x_0x_1'x_2x_3x_{15} + \\ & x_0x_1x_2'x_3'x_{16} + x_0x_1x_2'x_3x_{17} + \\ & x_0x_1x_2x_3'x_{18} + x_0x_1x_2x_3x_{19} \end{aligned}$$

01100010100100001000 $\rightarrow$ 0

What are the results like?— 2



What are the results like?— 3

Population at 5,000 problems in descending order of numerosity (first 40 of 77 shown).

	Condition	Action	PRED	ERR	FITN	NUM	GEN	ASIZ	EXPER	TST
0.	11 ## #0	1	0.	.00	884.	30	.50	31.2	287	4999
1.	00 1# ##	0	0.	.00	819.	24	.50	25.9	286	4991
2.	01 #1 ##	1	1000.	.00	856.	22	.50	24.1	348	4984
3.	01 #1 ##	0	0.	.00	840.	20	.50	21.8	263	4988
4.	11 ## #1	0	0.	.00	719.	20	.50	22.6	238	4972
5.	00 1# ##	1	1000.	.00	698.	19	.50	20.9	222	4985
6.	01 #0 ##	0	1000.	.00	664.	18	.50	23.9	254	4997
7.	10 ## 1#	1	1000.	.00	712.	18	.50	22.4	236	4980
8.	00 0# ##	0	1000.	.00	674.	17	.50	21.2	155	4992
9.	10 ## 0#	0	1000.	.00	706.	17	.50	19.9	227	4990
10.	11 ## #0	0	1000.	.00	539.	17	.50	24.5	243	4978
11.	10 ## 1#	0	0.	.00	638.	16	.50	20.0	240	4994
12.	01 #0 ##	1	0.	.00	522.	15	.50	23.5	283	4967
13.	00 0# ##	1	0.	.00	545.	14	.50	20.9	110	4979
14.	10 ## 0#	1	0.	.00	425.	12	.50	23.0	141	4968
15.	11 ## #1	1	1000.	.00	458.	11	.50	21.1	76	4983
16.	11 ## 11	1	1000.	.00	233.	6	.33	22.1	130	4942
17.	0# 00 ##	1	0.	.00	210.	6	.50	23.1	221	4979
18.	11 ## 01	1	1000.	.00	187.	5	.33	21.1	86	4983
19.	01 10 ##	1	0.	.00	168.	4	.33	19.1	123	4939
20.	11 #1 #0	0	1000.	.00	114.	4	.33	26.2	113	4978
21.	10 ## 11	0	0.	.00	152.	4	.33	23.9	34	4946
22.	10 1# 0#	1	0.	.00	131.	3	.33	21.7	111	4968
23.	00 0# 0#	0	1000.	.00	117.	3	.33	22.8	57	4992
24.	11 1# #0	0	1000.	.00	68.	3	.33	28.7	38	4978
25.	10 #1 0#	0	1000.	.00	46.	3	.33	20.6	4	4990
26.	10 ## 11	1	1000.	.00	81.	3	.33	23.9	113	4950
27.	#1 #0 #0	0	1000.	.00	86.	3	.50	23.6	228	4981
28.	01 10 ##	0	1000.	.00	61.	2	.33	22.5	16	4997
29.	01 00 ##	0	1000.	.00	58.	2	.33	22.2	46	4981
30.	10 0# 0#	1	0.	.00	63.	2	.33	22.8	22	4866
31.	11 0# #1	1	1000.	.00	63.	2	.33	23.2	35	4953
32.	00 1# #0	1	1000.	.00	77.	2	.33	20.7	7	4985
33.	10 #1 0#	1	0.	.00	93.	2	.33	24.5	28	4968
34.	11 #1 #1	1	1000.	.00	59.	2	.33	21.8	12	4983
35.	01 #1 #0	1	1000.	.00	75.	2	.33	23.1	21	4944
36.	01 #0 #1	0	1000.	.00	36.	2	.33	21.7	3	4997
37.	11 ## 01	0	0.	.00	92.	2	.33	19.7	41	4948
38.	10 ## ##	1	703.	.31	8.	2	.67	22.3	10	4980
39.	#1 1# #0	0	856.	.22	11.	2	.50	27.4	22	4978

Can you show the evolution of a rule?

Action sets [A] for input 101001 and action 0 at several epochs.

247

	Condition	Action	PRED	ERR	FITN	NUM	GEN	ASIZ	EXPER	TST
0.	## ## ##	0	431.	.440	8.	2	1.00	17.2	76	244
1.	## 10 ##	0	245.	.362	109.	2	.67	10.6	14	236
2.	## 10 0#	0	893.	.146	504.	5	.50	11.2	8	200

1135

	Condition	Action	PRED	ERR	FITN	NUM	GEN	ASIZ	EXPER	TST
0.	## #0 #1	0	519.	.419	1.	1	.67	16.5	11	1134
1.	## #0 0#	0	510.	.390	27.	2	.67	16.8	15	1119
2.	## 1# ##	0	125.	.261	0.	1	.83	21.7	18	1132
3.	#0 ## 0#	0	1000.	.021	4.	1	.67	17.7	0	1117
4.	#0 10 ##	0	454.	.433	2.	1	.50	14.8	53	1106
5.	#0 10 0#	0	735.	.343	27.	2	.33	14.4	13	1106
6.	1# ## #1	0	169.	.282	2.	1	.67	24.4	12	1119
7.	1# ## 0#	0	445.	.418	13.	5	.67	18.6	27	1119
8.	10 ## ##	0	1000.	.000	135.	2	.67	24.2	3	1117
9.	10 ## 0#	0	1000.	.000	451.	3	.50	23.4	17	1117

1333

	Condition	Action	PRED	ERR	FITN	NUM	GEN	ASIZ	EXPER	TST
0.	#0 1# 0#	0	761.	.336	1.	1	.50	10.6	10	1325
1.	1# ## 0#	0	652.	.387	5.	1	.67	10.9	11	1325
2.	1# #0 #1	0	107.	.197	6.	1	.50	22.0	8	1308
3.	1# 10 0#	0	829.	.228	26.	2	.33	14.3	9	1325
4.	10 ## 0#	0	1000.	.000	490.	4	.50	11.6	26	1325

2410

	Condition	Action	PRED	ERR	FITN	NUM	GEN	ASIZ	EXPER	TST
0.	1# ## 0#	0	360.	.394	0.	1	.67	18.1	14	2404
1.	10 ## 0#	0	1000.	.000	478.	10	.50	20.1	95	2392

2725

	Condition	Action	PRED	ERR	FITN	NUM	GEN	ASIZ	EXPER	TST
0.	#0 ## 0#	0	863.	.237	0.	3	.67	21.1	18	2714
1.	10 ## 0#	0	1000.	.000	630.	13	.50	22.6	117	2714
2.	10 #0 0#	0	1000.	.000	49.	1	.33	22.4	9	2638
3.	10 1# 0#	0	1000.	.000	58.	1	.33	18.4	8	2693

Why accurate, *maximally general* rules?

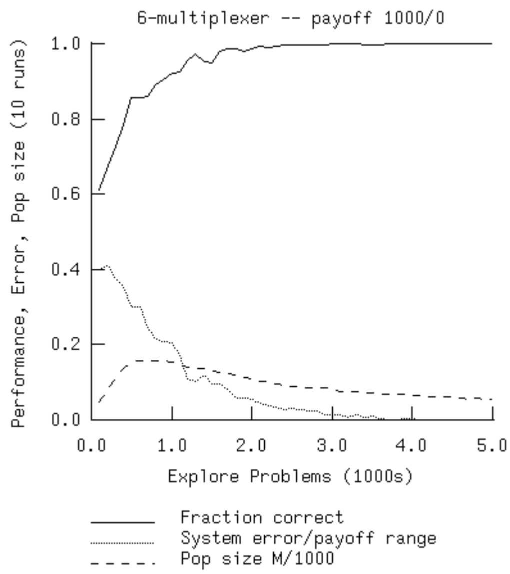
Consider two classifiers C1 and C2 having the same action, and let C2 be a generalization of C1. That is, C2 can be obtained from C1 by changing some non-# alleles in the condition to #'s. Suppose that C1 and C2 are equally accurate. They will therefore have the same fitness. However, note that, since it is more general, C2 will occur in *more action sets* than C1. What does this mean? Since the GA acts in the action sets, C2 will have *more reproductive opportunities* than C1. This edge in reproductive opportunities will cause C2 to gradually drive C1 out of the population.

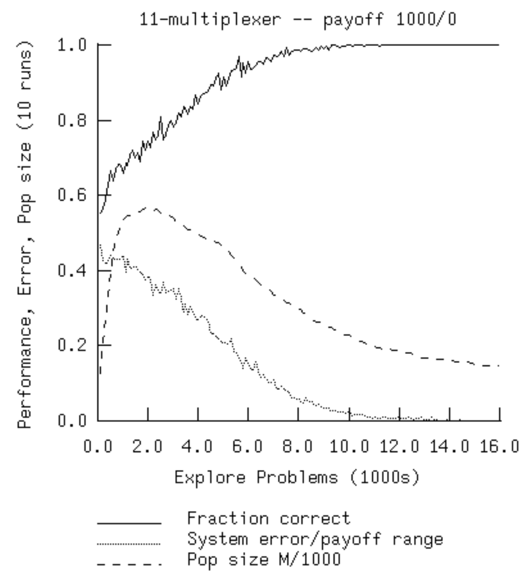
Example:			p	ε	F
C1:	1 0 # 0 0 1 : 0	$\Rightarrow$	1000	.001	920
C2:	1 0 # # 0 # : 0	$\Rightarrow$	1000	.001	920

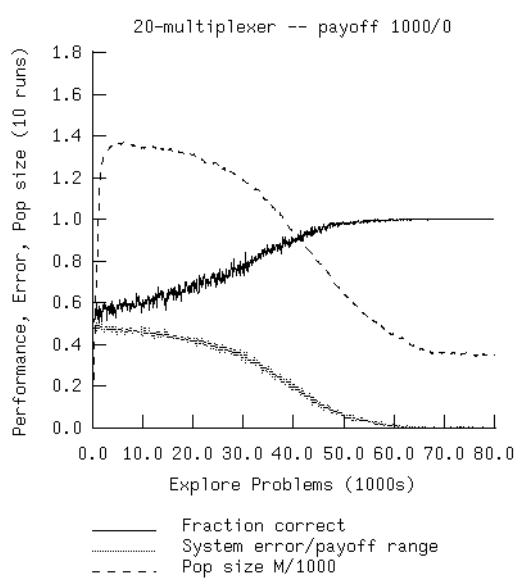
C2 has equal fitness but more reproductive opportunities than C1.

C2 will "drive out" C1

Does XCS scale up?







What about complexity?

20m ~5x harder than 11m
11m ~5x harder than 6m.

$\Rightarrow \; D = cg^p,$

where D = "difficulty", here learning time,
g = number of maximal generalizations,
p = a power, about 2.3
c = a constant about 3.2

Thus *"D is polynomial in g"*.

What is D with respect to l, string length?

For the multiplexers, $l = k + 2^k$,
or $l \rightarrow 2^k$ for large k.

But $g = 4 \cdot 2^k$, thus $l \sim g$,
So that "D is polynomial in l" (not exponential).

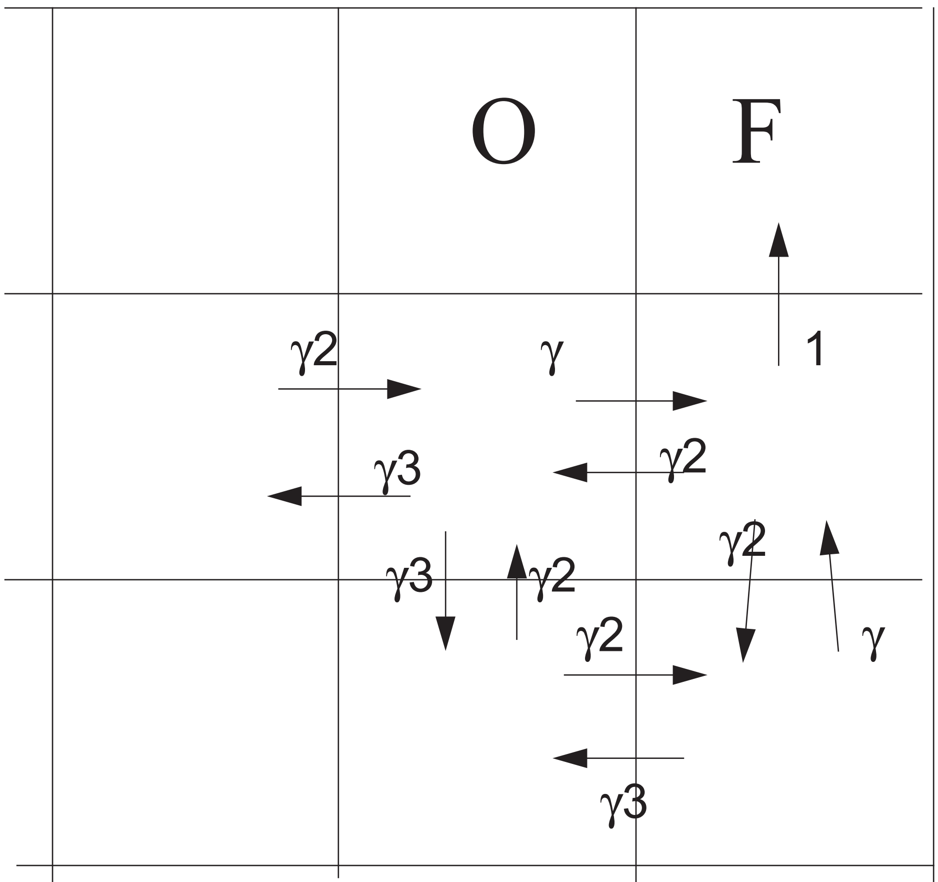
What about deferred reward?

Apply ideas from multi-step reinforcement learning.

Need the *action-value* of each action in each state.

What is the action-value of a state more than one step from reward?

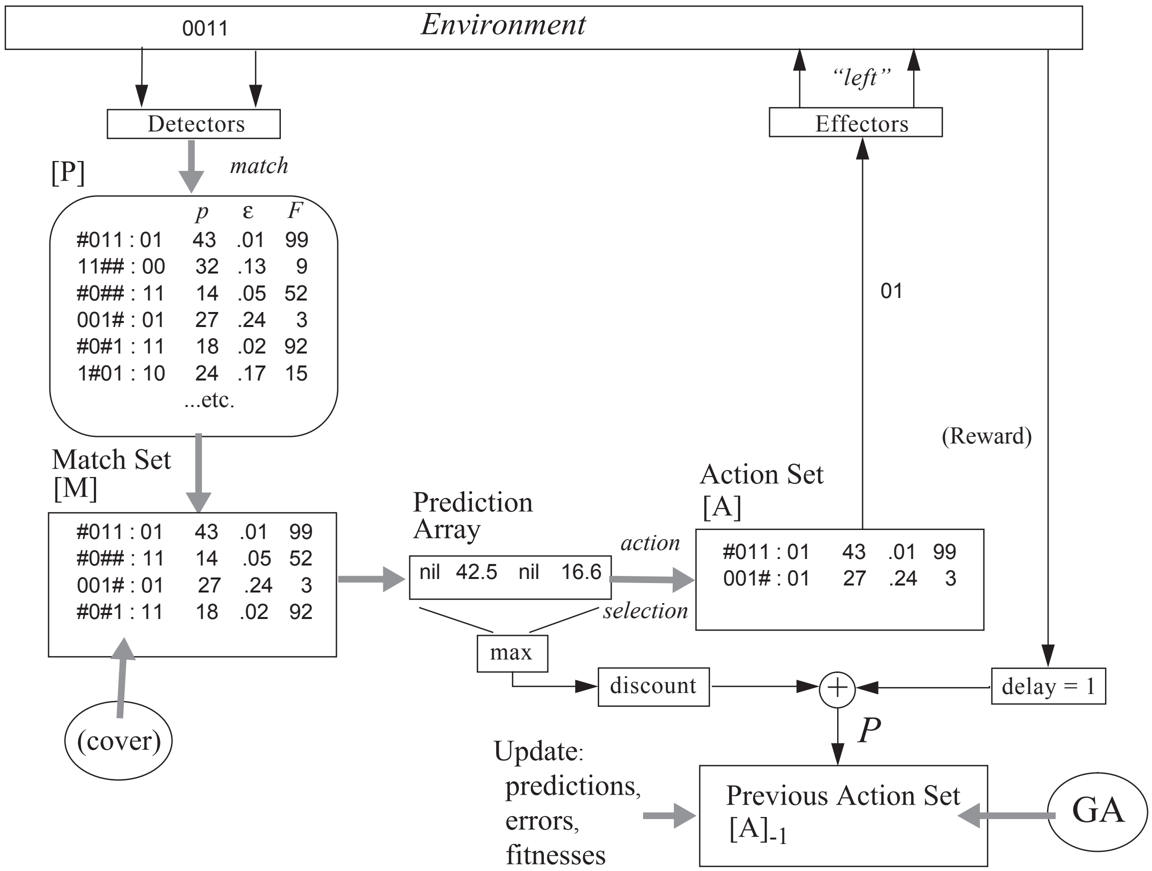
Intuitive sketch:



$$p_j \leftarrow p_j + \alpha[(r_{imm} + \gamma \max_{a' \in A} P(x',a')) - p_j]$$

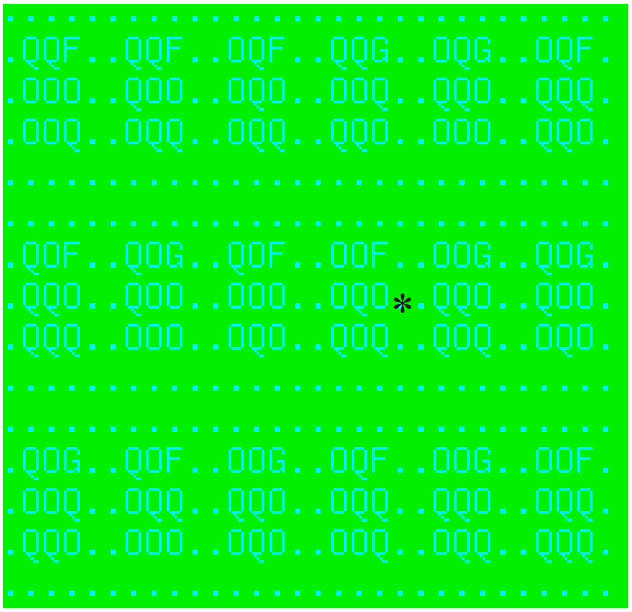
where p_j is the prediction of a classifier in the current action set [A],
x' and a' are the next state and possible actions,
$P(x',a')$ is a system prediction at the next state,
and r_{imm} is the current external reward.

Can I see the overall process?



- Previous action set $[A]_{-1}$ is saved and updates are done there, using the current prediction array for "next state" system predictions.
- On the last step of a problem, updates occur in [A].

What are the results like?— 1



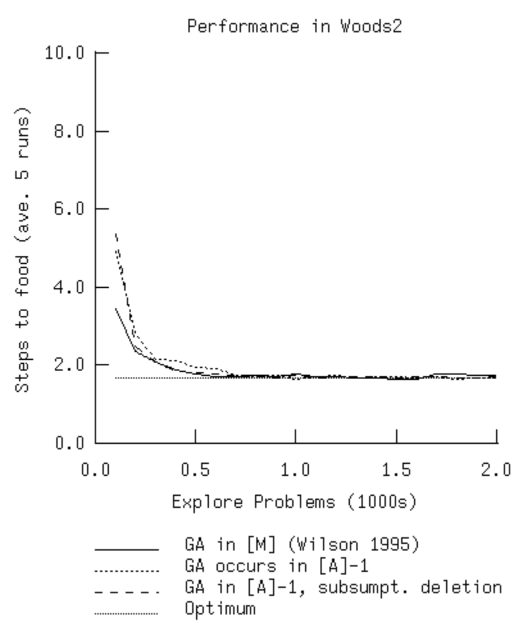
- Animat senses the 8 adjacent cells.

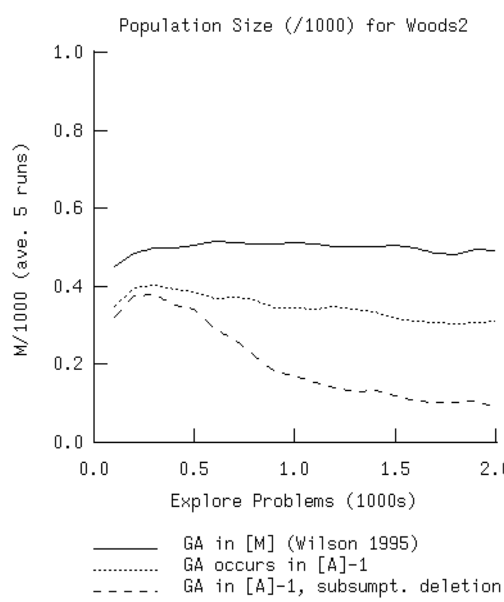
F b b
O * b
Q b b

- Coding of each object:

F = 110 “food1”
G = 111 “food2”
O = 010 “rock1”
Q = 011 “rock2”
b = 000 “blank”

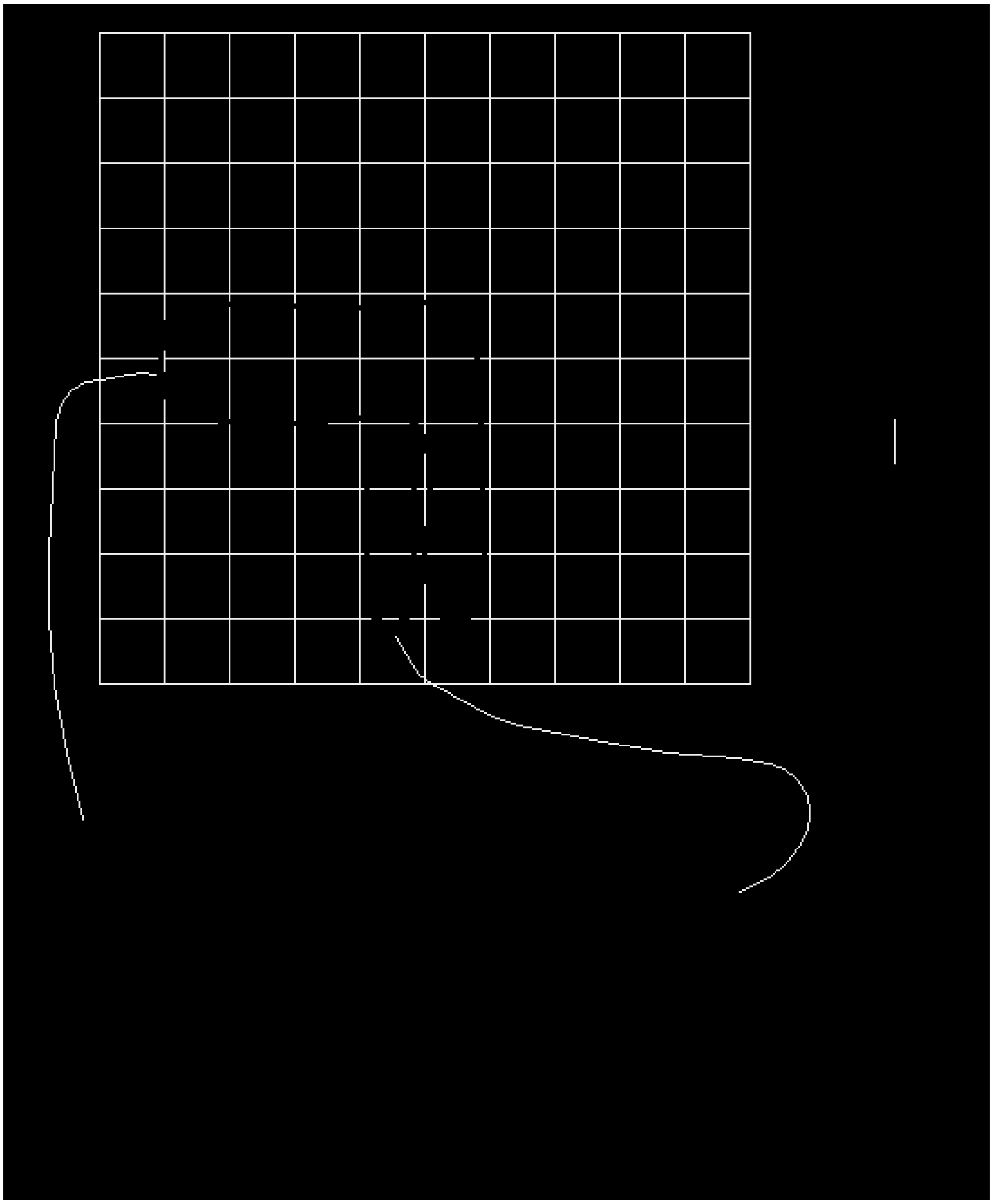
- “Sense vector” for above situation: 000000000000000011010110
- A matching classifier: ####0#00####00001##101## : 7





What are the results like?— 2

Two generalizations discovered by XCS in Woods1.



What about continuous inputs, actions, time?

Inputs:

$<<x_1 \pm \Delta x_1> \ldots <x_n \pm \Delta x_n>> : <\text{action}> \Rightarrow p$

Actions:

$<<x_1 \pm \Delta x_1> \ldots <x_n \pm \Delta x_n>> : <a \pm \Delta a> \Rightarrow p$

—and combine matching rules à la fuzzy logic, perhaps.

Time:

$<<x_1 \pm \Delta x_1> \ldots <x_n \pm \Delta x_n>> : <a \pm \Delta a> \Rightarrow \frac{dp}{dt}$

—action selection based on steepest ascent of p.

What about Non-Markov environments?

Example (McCallum's Maze):

0	0	0	0	0	0	0
0		*		*		0
0		0		0		0
0		0	F	0		0
0	0	0	0	0	0	0

* Aliased states. Optimal action not determinable from current sensory input.

Approaches:

- "History window" — remember previous inputs
- Search for correlation with past input events
- ✓ Adaptive internal state:

Environmental condition External action Prediction

#1 ## 0# ## ## ## # : 1 0 $\Rightarrow$ 504

Internal condition

Internal action

What if generalizations are not conjunctive?

Example: "if $x > y$ for any x and y, and action a is taken, payoff is predicted to be p."

Cannot be represented using a single classifier with traditional conjunctive condition, since it's a relation.

However, it can be represented using an "s-classifier":

$(> x\ y)$: <action a> $\Rightarrow$ p

i.e., a classifier whose condition is a Lisp s-expression.

With appropriate elementary functions, s-classifiers can encode an almost unlimited variety of conditions.

They can be evolved using techniques of genetic programming.

How is XCS different from other RL systems?

Rule-based, not PDP ("parallel distributed processing")

- Structure is created as needed
- Learning may often be faster because classifiers are inherently non-linear
- Learning complexity may be less than most PDPs
- Classifiers can keep and use statistics; difficult in a network
- Hierarchy and reasoning may be easier, since knowledge is in subroutine-like packages