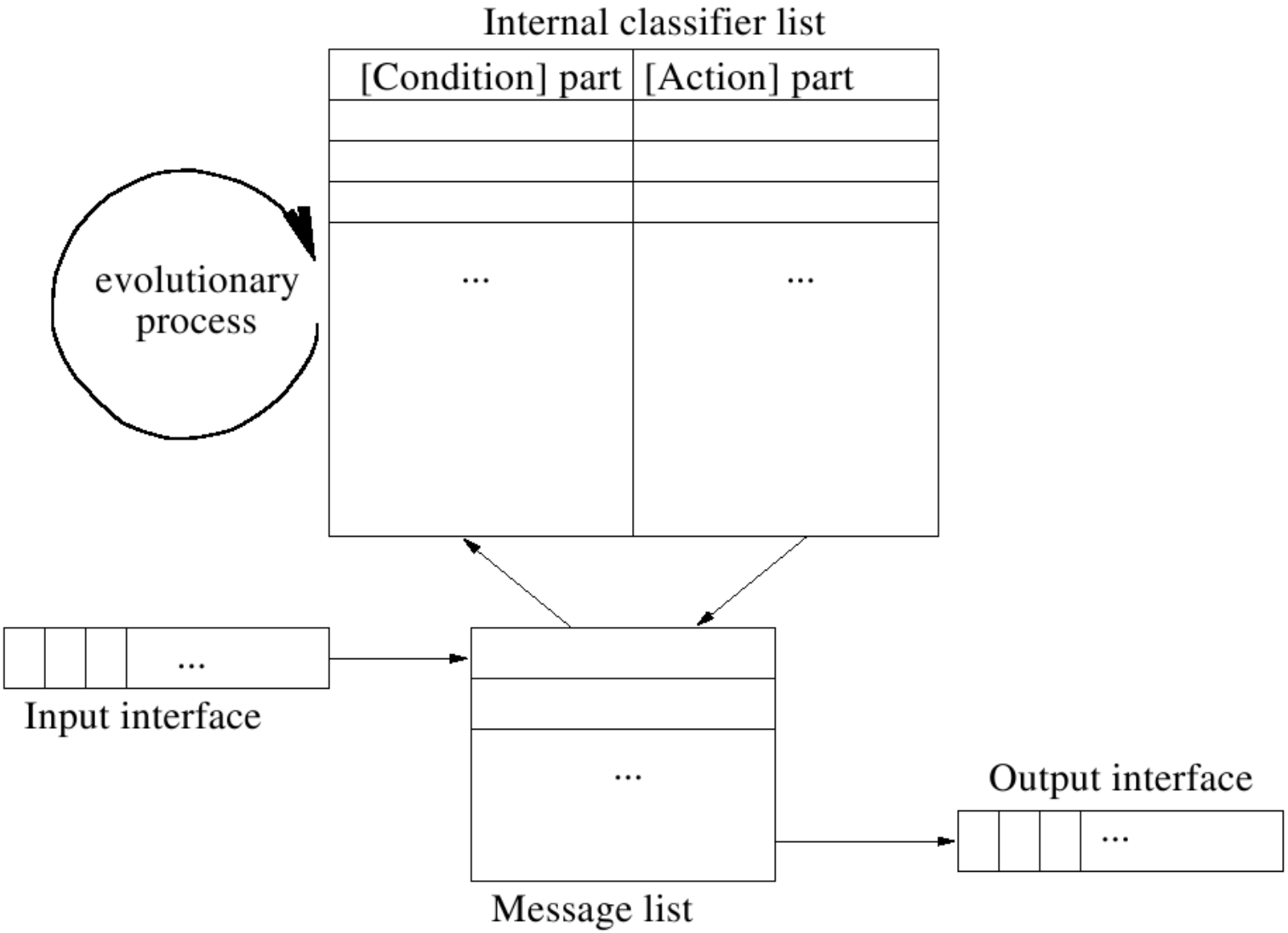
Modular Evolutionary Approaches

Learning Classifier Systems

(*"Learning Classifier Systems: A Survey"* Olivier Sigaud, Stewart W. Wilson)
(Jan Drugowitsch *"Design and Analysis of Learning Classifier Systems: A Probabilistic Approach"*, Springer 2008)

Pittsburgh versus Mitchigan approach, CS1

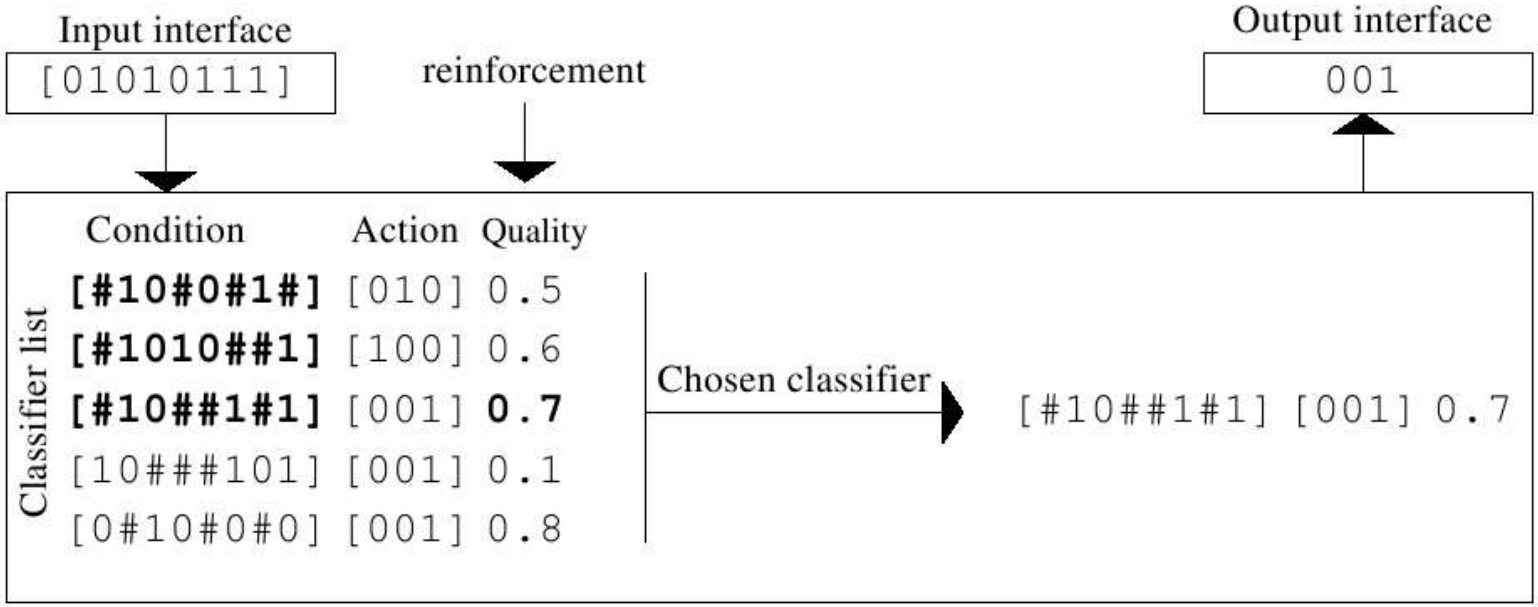
- LCSs were invented by Holland (Holland, 1975) in order to model the emergence of cognition based on adaptive mechanisms. They consist of a set of rules called classifiers combined with adaptive mechanisms in charge of evolving the population of rules.
- In Smith's approach (Smith, 1980), from the University of Pittsburgh, the only adaptive process was a GA applied to a population of LCSs in order to choose from among this population the fittest LCS for a given problem.
- By contrast, in the systems from Holland and his PhD students, at the University of Michigan, the GA was combined since the very beginning with an RL mechanism and was applied more subtly within a single LCS, the population being represented by the set of classifiers in this system.


Internal classifier list
[Condition] part
[Action] part
...
...
evolutionary process
...
Input interface
...
Output interface
...
Message list

Strength-based LCS ZCS

- Wilson simplified the CS1 by removing the message list: ZCS
- estimation of the accumulated reward of a classifier is also its GA-fitness
- the action is selected from the match-set $[M_t]$ of classifiers matching the input at time t with probability proportional to their strength
    - *don't care* symbols # match anything; interval matching for real number vectors
- reward estimation is closer to SARSA than to Q-LEARNING
    1. the action-set $[A_{t-1}]$ of classifiers from $[M_{t-1}]$ which advocated the action a_{t-1} chosen at the previous time step share equally a fraction $\gamma\,\alpha$ of the sum of the values of classifiers in $[A_t]$;
    2. all classifiers in $[A_t]$ share equally a fraction α of the reward r_t received for executing a_t;
    3. the value of all classifiers in $[M]$ which do not belong to $[A_t]$ is reduced with a tax τ

- rule creation
  - the GA selects two random parents proportionally to their fitness, performs one-point crossover and perhaps mutation, sets the initial fitness as the parent's average, replaces two random rules inversely-proportionally to their fitness
  - when $[M_t]$ is empty or has only low-fitness rules, the covering operator creates a rule matching current input with a random action and initial fitness equal to the average fitness of the population
- the main drawback of ZCS are overly general rules which can lead to sub-optimal behavior



Accuracy-based LCS XCS

- action selection uses any of the exploration-exploitation mechanisms from RL; classifiers (rules) can be evaluated individually or grouped into prediction arrays of rules voting for the same action
- classifier evaluation close in spirit to Q-LEARNING:
  - expected reward p given immediate reward r: $p \leftarrow p + \beta(r - p)$
  - prediction error: $\epsilon \leftarrow \epsilon + \beta(|r - p| - \epsilon)$
  - raw accuracy: $k = \begin{cases} 1 & \text{if } \epsilon < \epsilon_0 \\ \alpha\left(\frac{\epsilon}{\epsilon_0}\right)^{-\nu} & \text{otherwise} \end{cases}$ where $\nu > 0$
  - relative prediction accuracy k' with respect to $[A_t]$
  - fitness: $f \leftarrow f + \beta(k' - f)$
- the genetic operators select parents from $[A_t]$, and delete random classifiers proportionally to the sizes of their action sets, estimated by updates $a_s \leftarrow a_s + \beta(|A_t| - a_s)$ for $a_s \in A_t$ – generalization ballance
- standard and guided mutation: not leaving the action set $[A_t]$
- subsumption: if a classifier more general than a newly created one is fit, increase its numerosity instead of adding the new one

- extensions to XCSs:
  - use EDA as the GA to model the structure of the condition features
  - deal with non-Markovian environments perceptual aliasing problem
    - **classifier chaining** for actions leading to aliased states
    - **internal register management**: memory matched by conditions and modified by actions
  - XCSF, a generic function approximator "mixture of localized experts"

Anticipation-based LCS ACS

(*Combining Latent Learning with Dynamic Programming in the Modular Anticipatory Classifier System* Pierre Gerard, Jean-Arcady Meyer, Olivier Sigaud)

- anticipation-based systems have `[condition][action]-->[effect]` rules, they build a model of transitions
- action selection in the MACS system:
    1. priority for actions bringing more information for untried transitions
    2. then, actions bringing more external reward
    3. for actions equivalent wrt. above conditions, those that haven't been tried for the longest time
- ACS and YACS have a *don't change* symbol `=` that copies an attribute from the `condition` to the `effect`; MACS has a *don't know* symbol `?`, an `effect` counterpart of the `condition` symbol `#`
- MACS updates the good/bad anticipation counts for every classifier at each step (using the $\langle s_{t-1}, a_{t-1}, s_t \rangle$ triple)

- the expected improvement by specialization measures (using the "delta update") for each *don't care* attribute whether the good/bad current anticipation could be separated from the last bad/good anticipation by specializing this symbol; oscillating classifiers are specialized using it
- similarily the expected quality of generalization measures for each specific condition attribute, for classifiers whose action match a_{t-1} and condition doesn't match s_{t-1} but would match if the corresponding attribute were generalized, if it would make a good or bad anticipation
    - a minimal set of classifiers covering all anticipations is selected when adding generalizations
- when no classifier matches the current $\langle s_{t-1}, a_{t-1} \rangle$, new ones anticipating parts of s_t are created with the most general conditions matching s_{t-1} and still different from other classifiers with the same action and effect
- the immediate information gain measures how much a transition helps in directing rules into deletion, specialization or generalization; it acts as an internal reward
    - iterative planning is used to have a "deep" explorative policy
- MACS is cautious: $Q(s, a)$ is computed by minimization over anticipations

Baum & Durdanovic Artificial Economy

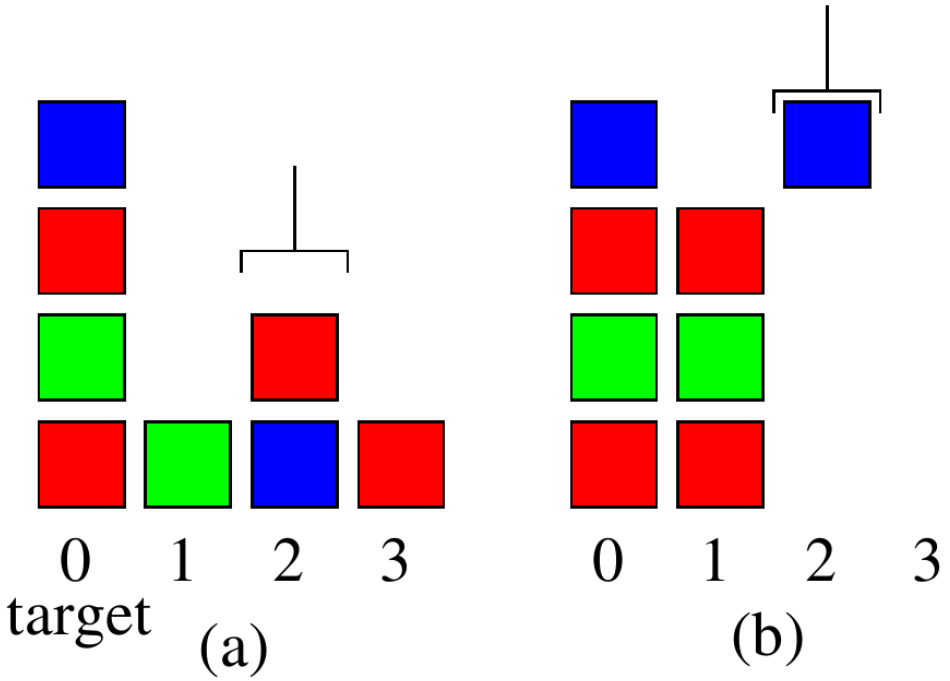
Hayek Evolutionary Economy Systems

(*An Evolutionary Post Production System* Eric Baum, Igor Durdanovic)

- starts from the same inspiration as recent LCSs: J. Holland's CS1
- fixes the **bucket brigade** credit assignment algorithm into an economic model with *conservation of money* and *strong property rights*
- uses a more powerful agent language (Turing-complete in Hayek4), more expressive under the "dynamically stable configurations" restriction
- Hayek4 evolved a program solving **arbitrary** Blocks World instances, where
  - TD-learning solved 2 blocks deep instances,
  - TD with hand-coded features – 8 blocks deep instances,
  - Genetic Programming with hand-coded features solved a single 9 block deep instance
  - GP with a powerful hand-coded feature *number correct* – 5 blocks deep instances
  - Inductive Logic Programming – 2 blocks deep instances

- Hayek4 evolves a population of agents each being a variant of string rewriting system (Post system); (actually, the RHS are encoded as actions)
- Computation proceeds in a series of auctions, the actions of the highest bidder are applied, the winning agent pays its bid to the winner of the previous auction. If it solves the instance and says done, it collects reward from the world.
- After each instance, all agents are assessed a tax proportional to the amount of computation they have done (evolutionary pressure), any agent with less money than it was initiated with is removed and its money returned to its creator.
- Any rule not used in some living agent, and any rule that has not been applicable in the last 1000 instances is removed.
- (W grows to be slightly more than the currently biggest reward.) Any agent with wealth at least $10W$ creates a (mutated) child, giving it an initial endowment of W.
- A special agent *Root* does not bid but simply creates random agents.
- At the end of each instance, each agent passes .25 fraction of its profit in that instance to its parent.

- The first time a new agent has a production that matches, the agent is assigned a bid that is ε higher than all competing bids. The new agent thus wins that auction, and its bid is then fixed.
- Other systems (e.g. CS1, ZCS) violate property rights by dividing the reward among a set of active agents, or selecting agents probabilistically (proportionally to their bids).



The task of the Blocks World problem is to stack the blocks from stacks 1-3 on stack 1 according to the pattern (stack 0).

NeuroEvolution by Cooperatively Coevolved Synapses

(*Accelerated Neural Evolution through Cooperatively Coevolved Synapses* Faustino Gomez, Juergen Schmidhuber, Risto Miikkulainen)

- In cooperative coevolutionary algorithms the species represent solution components.
-