Flexible Shaping: How learning in small steps helps

Hierarchical Organization of Behavior, NIPS 2007

Kai Krueger and Peter Dayan

Gatsby Computational Neuroscience Unit

Outline

- Introduction
  - learning may require external guidance
  - shaping as a concept:
- Set-Up
  - 12-AX task, LSTM network, shaping procedure
- Results
  - simple shaping
  - when does shaping help most?
  - flexibility to adapt to variations
- Conclusions, issues and future work
  - rules and habits

Introduction

- Learning essential for flexibility
  - trial and error
  - external guidance:
    - “one shot teaching” by verbal explanation of abstract rules
    - imitation
    - shaping
- Guidance critical for complex behavior
  - branching, working memory, rapid changes

Shaping

- “a method of successive approximations”
  (Skinner 1938)
- Key features:
  - external alteration of reward contingencies
  - withdrawal of intermittent rewards
- Creates behavioral units
  - e.g. lever pressing of a rat
- Separate time scales / branching points
  - by providing separate stages in shaping
- Ubiquitously (and implicitly) in animal experiments

12-AX task

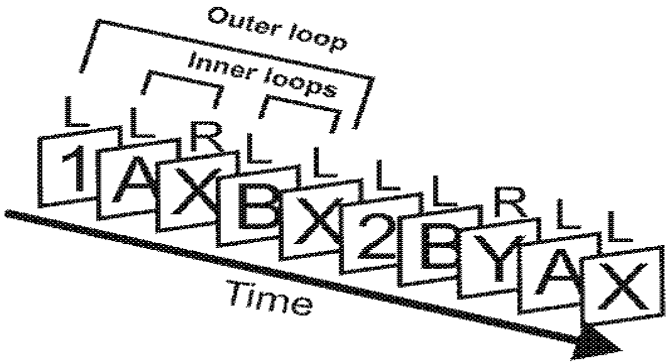
Demo

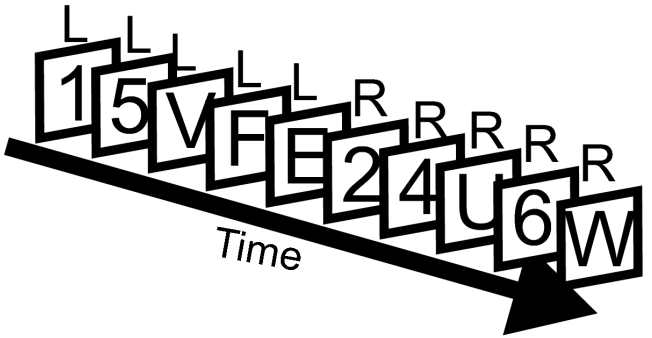
LSTM network

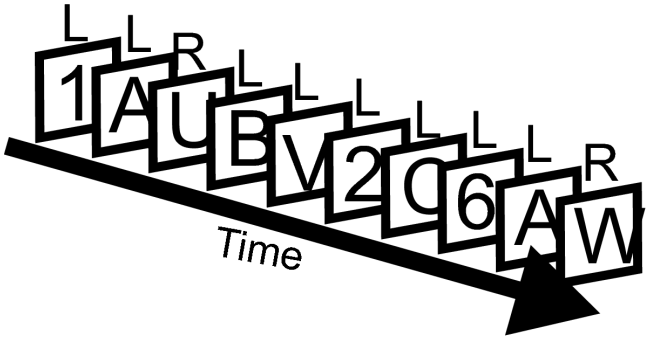
- Long Short-Term memory *(Hochreiter and Schmidthuber 1997)*
  - *3-layer recurrent neural network*
- Provides built-in mechanisms for:
  - working memory
  - gating (input, output and forget)
- Abstract “over-simplified” model of PFC
  - basis to motivate PBWM (O’Reilly et al.)

Shaping procedure

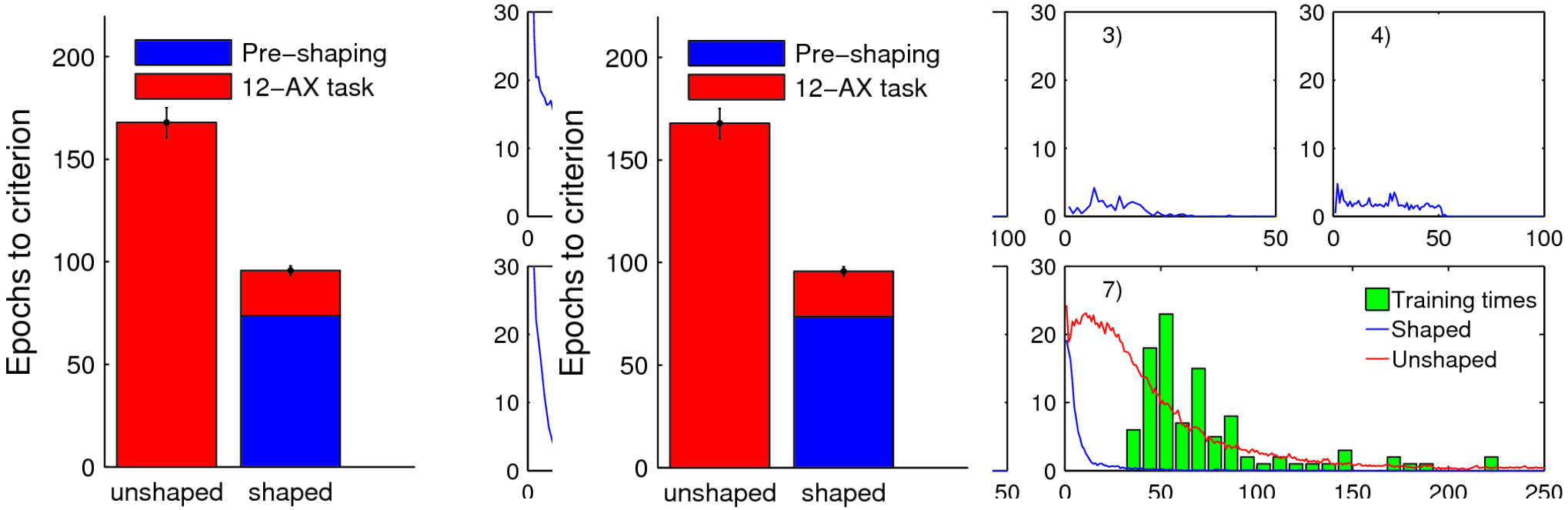
- Teach 12-AX as successive approximations
- Separate WM timescales:
  - long: (1 / 2)
  - short: (AX/BY)
- Learning in 7 stages
  - last stage: full 12-AX
- Resource allocation
  - currently done by hand
  - each stage learned into a new block
    - all other memory blocks disabled
  - provides separation / No interference







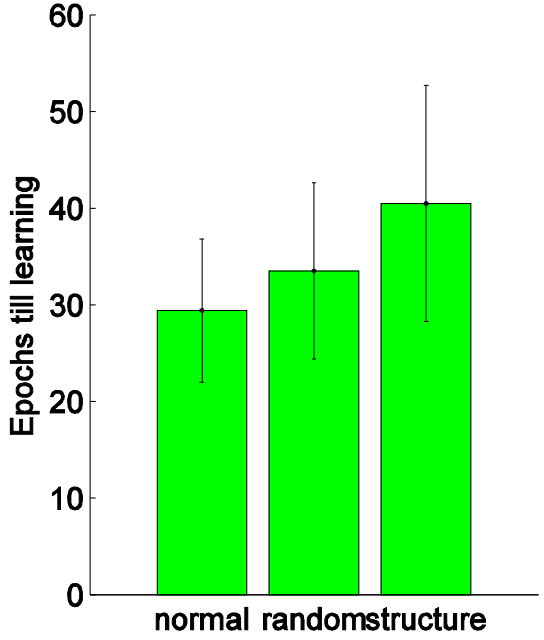
Simple shaping

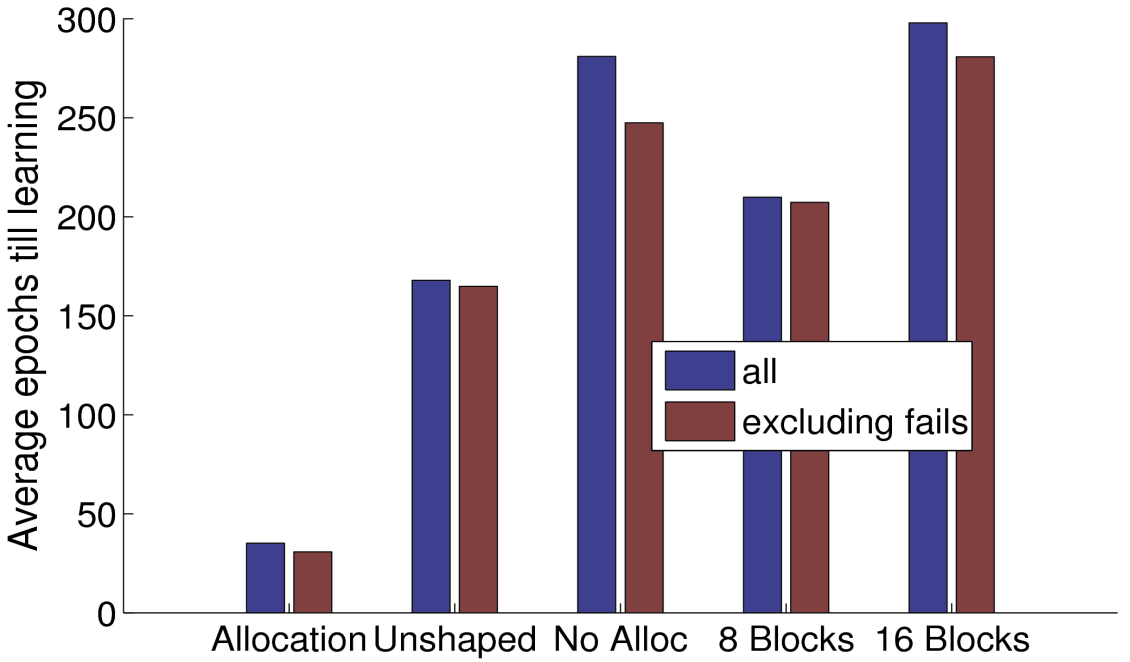


- Improvement in learning times:
  - 8 fold decrease (only final stage)
  - significantly better (including complete training)
  - median: 13 epochs, min: 8 epochs
- Need the 4 stages of shaping 1 and 2
- High variance in shaping times

What makes shaping work

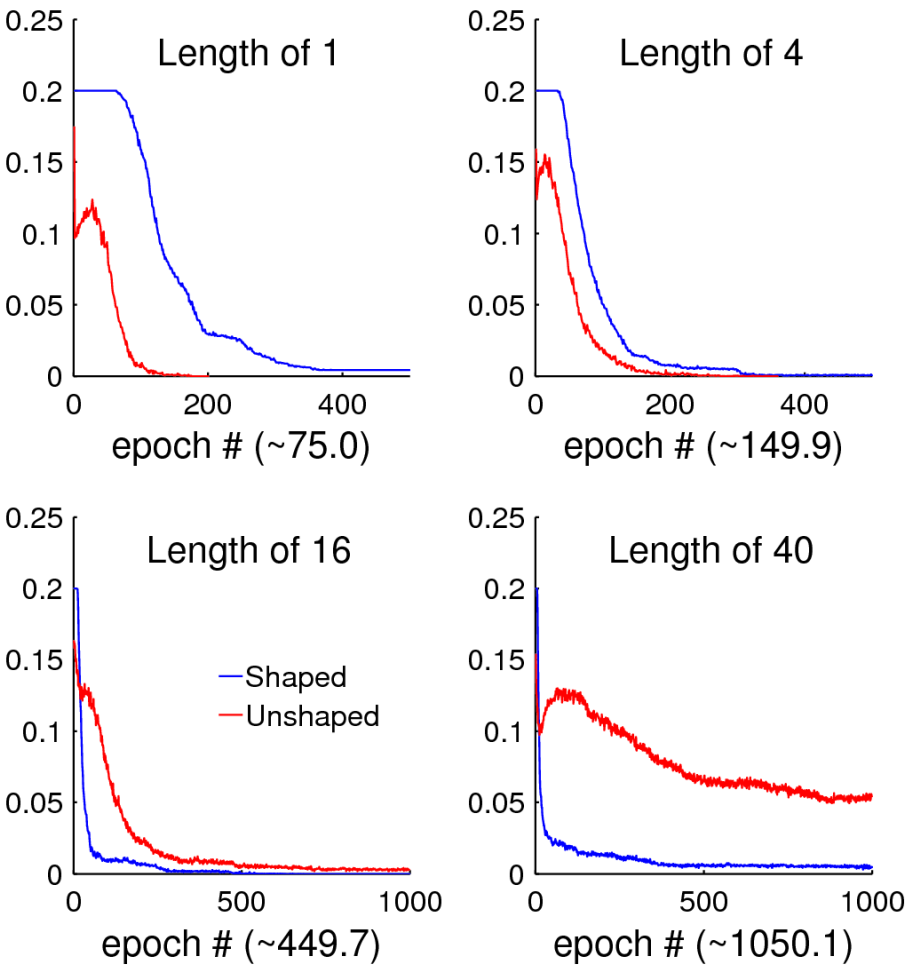
- Robustness to additional structure:
  - irrelevant “experience”
    - related and unrelated tasks / inputs
- Resource allocation:
  - interference between tasks => no benefits





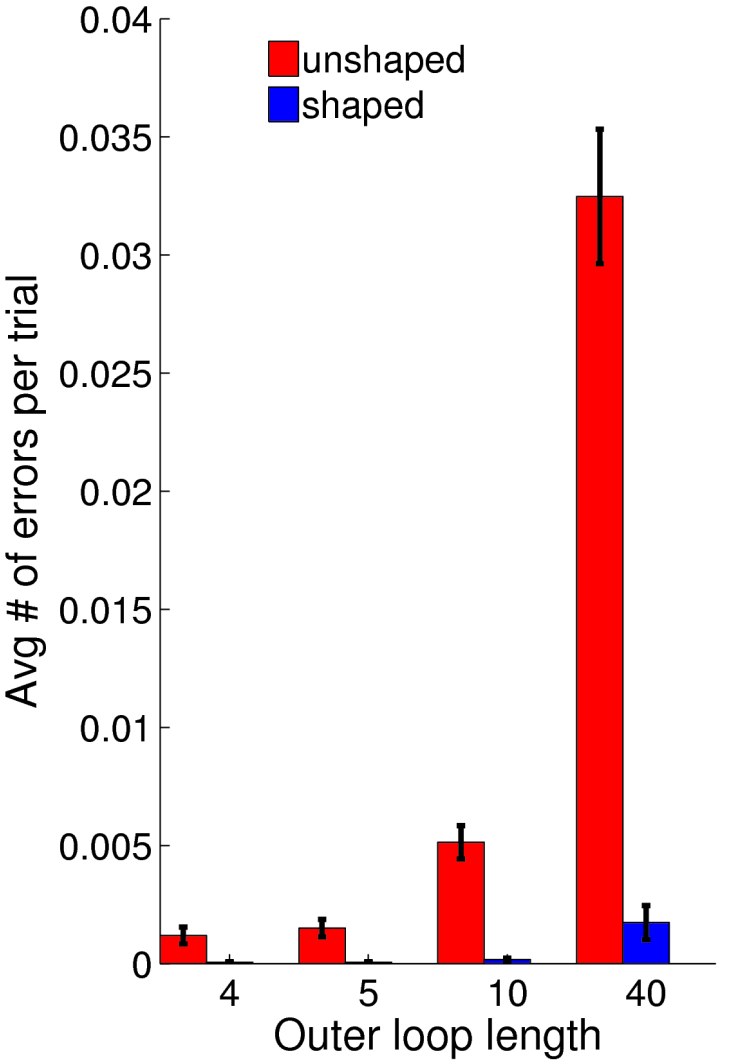
Shaping: when is it useful?

- Can shaping prevent scaling of learning time with task complexity?
- One aspect of complexity: Temporal credit assignment
  - increase the outer loop length => higher temporal complexity
- Results:
  - training time still increases, but scales much slower.
  - increasing complexity => shaping more important



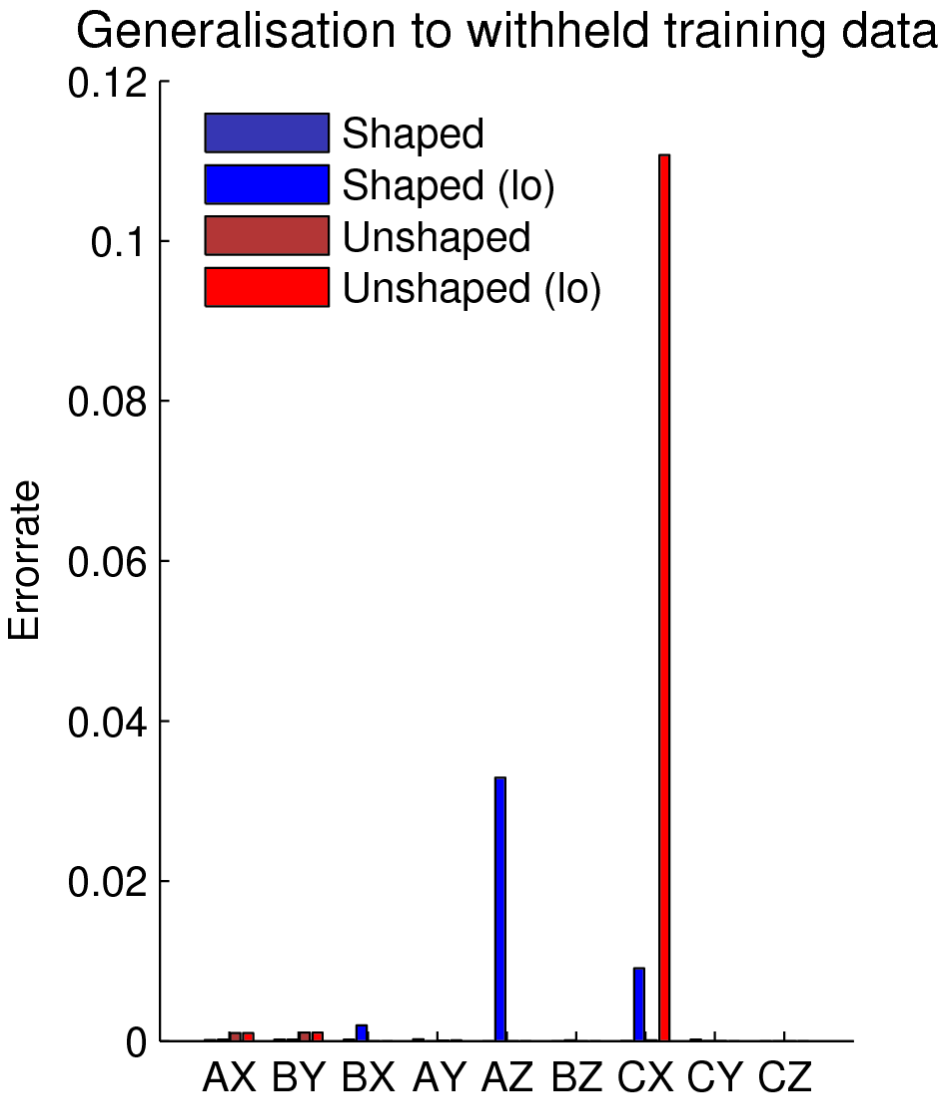
Rule abstraction

- Rule abstraction:
  - flexibility to cope with change in statistics
- Train on the base 12-AX task (loop length 4)
- Test with variations
  - loop lengths 4, 5, 12, 40
  - disable learning
- Should perform perfectly
  - abstract rules have not changed

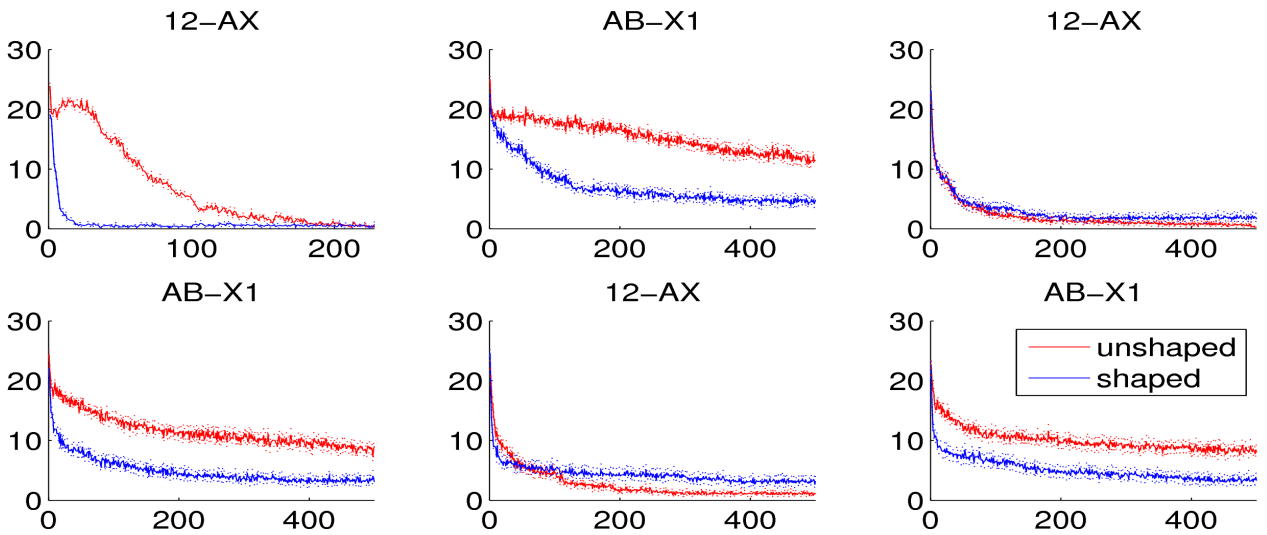


Generalisation

- Generalisation:
  - cope with yet unseen data (inner loop combinations)
- Train 12-AX task without AZ and CX
- Test performance on full task
- Only 7 combinations
  - one valid generalisation only?
- Mixed results:
  - differences in emphasis (1-back / 0-back)
  - overall shaping still better



Reversal learning



- Reverse stimulus – rule association
  - shape all components needed
- Repeatedly reverse (after 500 epochs)
  - learning of reversals.
- Identify flexibility to perform reversals
  - unshaped: mostly fails
  - shaped: succeeds more often

Conclusions

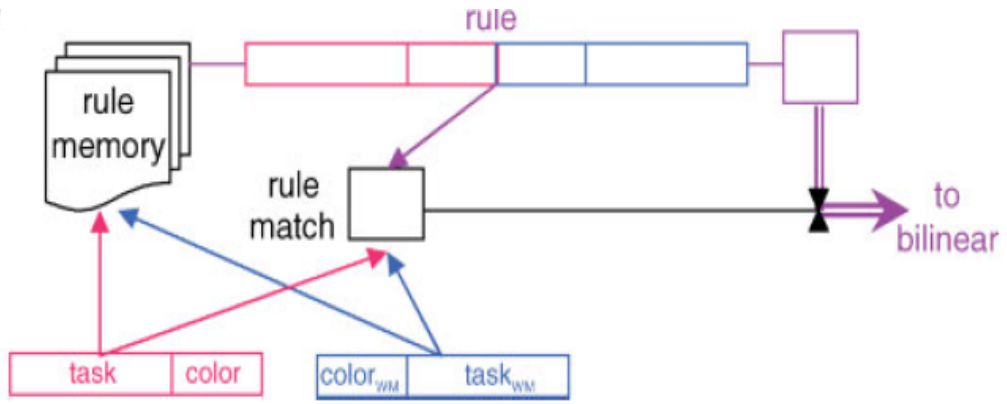
- Shaping works
- Reduces learning times
- Helps learning long time delays
  - separating time scales of actions
  - recombine “behavioral units” into sequences
- Improves abstraction and separation
- Increases flexibility to reversals
- Take home message:
  - need to take sequential and transfer learning more into account when looking at learning architectures.
- Still issues to solve though

Limitations

- Resource allocation
  - prime computational issue
    - done by hand (Homunculus)
  - ideas to automate:
    - compute “responsibilities”
    - Mosaic
- Experimental data
  - no published data on learning 12-AX
  - interesting manipulations:
    - loop length, target frequency, …
    - natural grouping of alphabet

Future Work

- Still based on “habitual” learning => no instant reprogramming
- Need additional mechanisms:
  - more explicit rules
  - variable substitution
- Bilinear rules framework: (Dayan 2007)
  - recall
  - match
  - execute



- Close interaction between habitual and rule based learning
  - rules supervise habit learning
  - habits form basis of rule execution
- Results in a task grammar?

Questions?

Thank you