Biologically-Inspired Control in Problem Solving

Thad A. Polk, Patrick Simen,
Richard L. Lewis, & Eric Freedman

Computational Models of Control

- Challenge: Develop computationally explicit theories of control deficits in complex problem solving
  - Where control deficits are often most apparent
- Symbolic models (production systems)
  - E.g., ACT-R, Soar, Epic, ...
  - Natural model of flexible, goal-driven behavior and therefore easier to apply to complex problem solving
  - But harder to map onto the brain and patients
- Neural networks
  - E.g., Cohen, Levine, Dehaene, Braver, O'Reilly, ...
  - Neural mechanism for control (modulation) and therefore easier to map onto the brain and patients
  - But harder to apply to complex problem solving

Major Points

1. Natural & explicit mapping from goal-driven production systems onto neural computation
   – Makes it possible to build plausible neural models of complex problem solving

2. Mapping leads to explicit hypothesis about the role of DLPFC in problem solving:
   – Represents internally generated subgoals that modulate among choices

3. Applying to TOL accurately simulates human behavior
   – Intact model simulates normals, even on hardest problems
   – Lesioning subgoal net simulates prefrontal deficits

Plan

- Symbolic models of control
- A simple model of neural computation
- Mapping symbolic control onto neural nets
- Network model of Tower of London
  - Intact behavior
  - Damaged behavior

Goal-Driven Production Systems

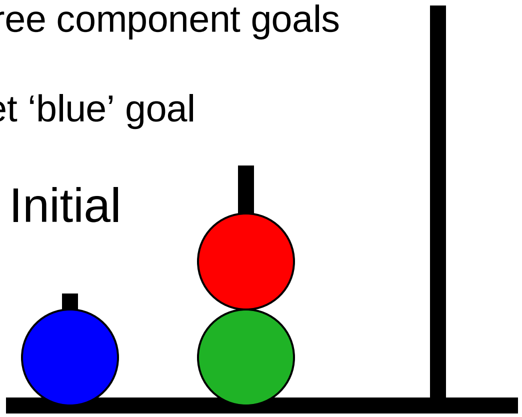
- Almost all models of complex cognition based on production systems (ACT-R, Soar, Epic)
  - Set of symbolic IF-THEN rules that match against memory and take actions and/or change memory:
- Production systems proposed as control theory (Newell, 1973):
  - Allow behavior to be flexible, opportunistic, interruptible
  - Next step chosen dynamically based on what's in memory/world
  - Goals/subgoals modulate/control decisions

Example

Decompose problem into three component goals

Set ‘blue’ goal

Initial

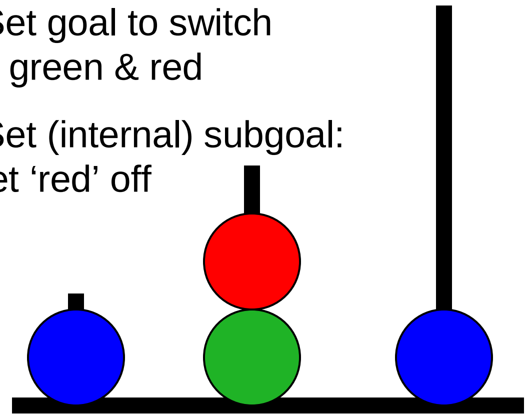


•Data-driven productions: Consider both legal moves

•(External) Goal-driven control: Pick move to achieve ‘blue’ goal

•Set goal to switch green & red

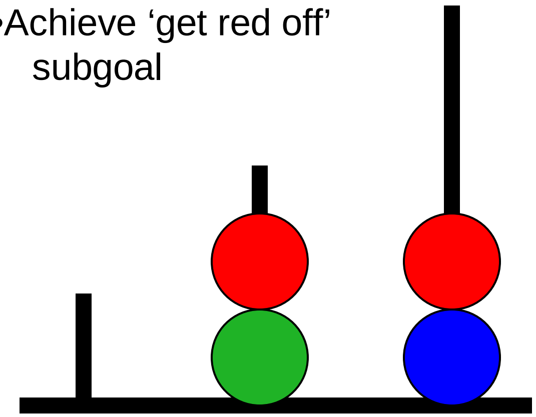
•Set (internal) subgoal: get ‘red’ off



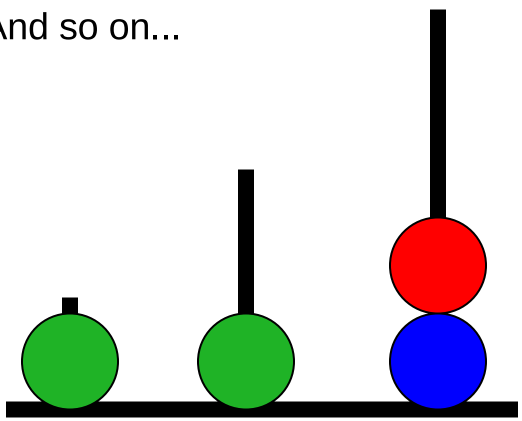
•Data-driven productions: Consider all legal moves

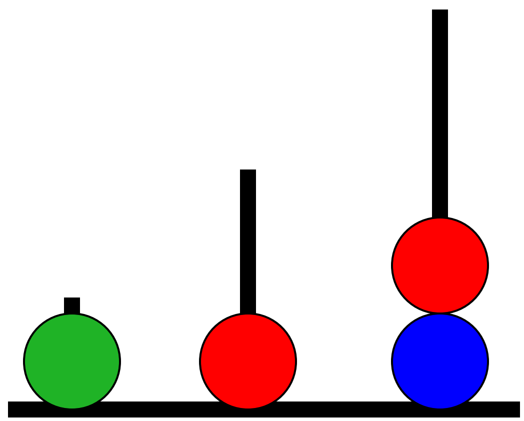
•(Internal) Goal-driven control: Pick move to ‘get red off’

•Achieve ‘get red off’ subgoal

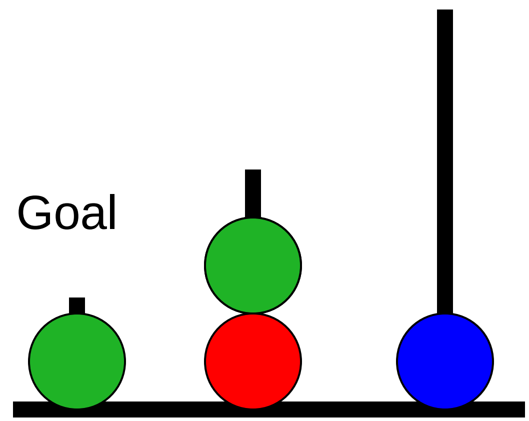


•And so on...





Goal



Key Features of Symbolic Control

- Data-driven production rules
  - Asymmetric associations between symbols
  - Allows flexible behavior that reacts to current state
  - E.g., Recognizing legal moves in current TOL state

- Top-down control from current goal/subgoal
  - Constrains data-driven processing
  - Both external goals and internally generated subgoals can control
  - E.g., preferring moves that satisfy current subgoal over others

Plan

- Symbolic models of control
- A simple model of neural computation
- Mapping symbolic control onto neural nets
- Network model of Tower of London
  - Intact behavior
  - Damaged behavior

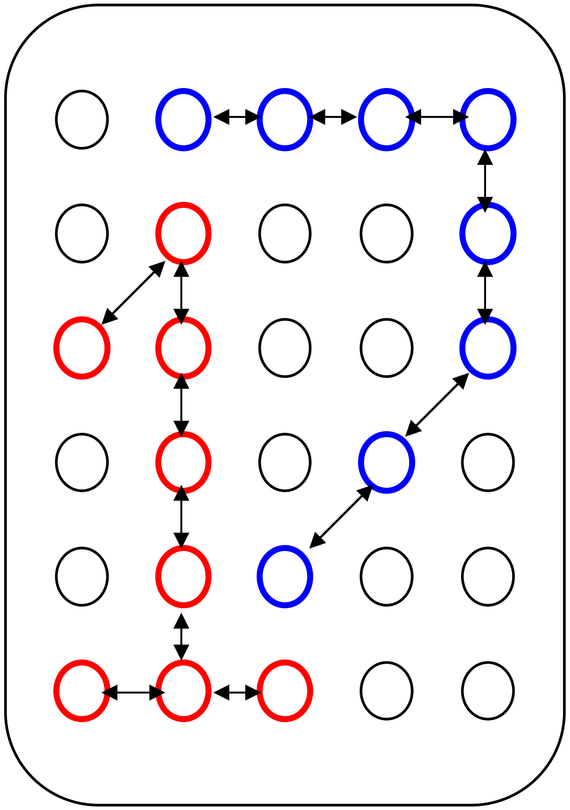
Neural Computation: Assumptions for Present Model

1. Neural processing is *recurrent*

2. Neural representations are *distributed*

3. Neural learning is *correlation-based* (Hebbian)

Emergent Property:
Attractors (Discrete Stable States)
(Hopfield, 1982; 1984)

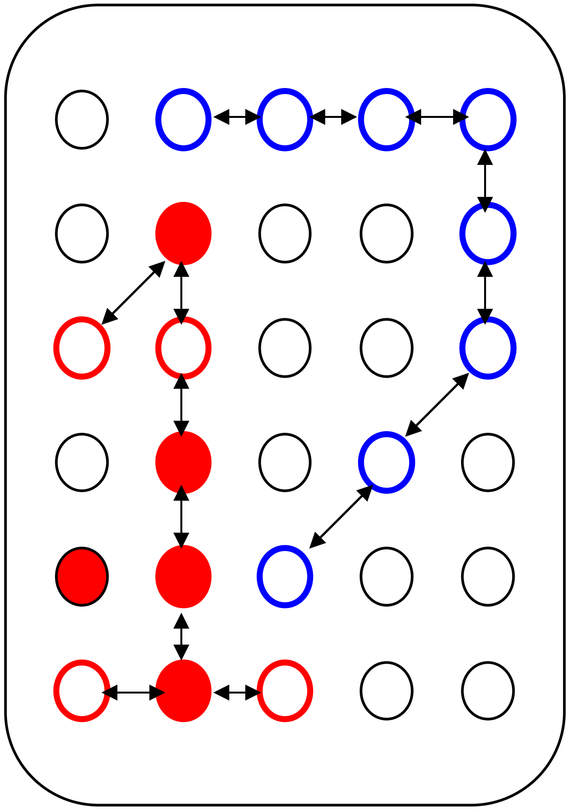


If distributed patterns occur frequently, they become discrete stable states for the network...

…and the network will converge on them given any similar patterns...

Emergent Property:
Attractors (Discrete Stable States)

(Hopfield, 1982; 1984)

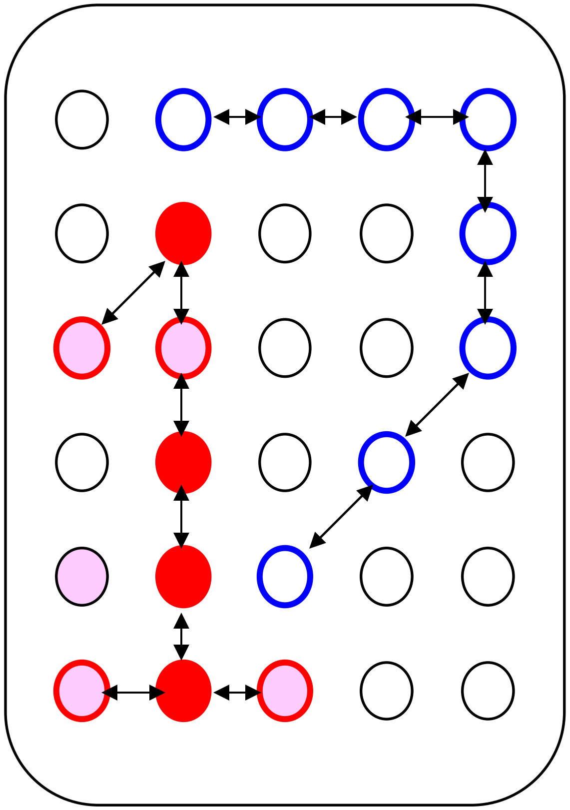


If these patterns occur frequently, then they become discrete stable states for the network.

And the network will converge on them given any similar patterns...

Emergent Property: Attractors (Discrete Stable States)

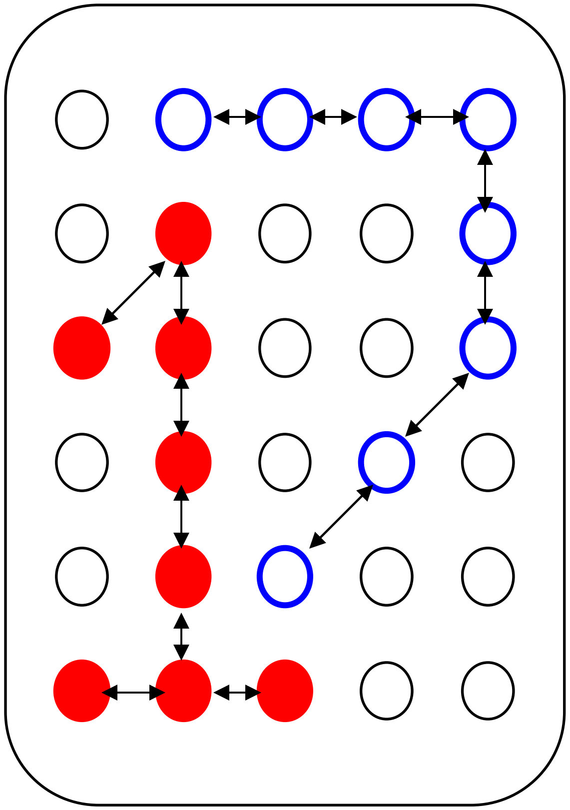
(Hopfield, 1982; 1984)



If these patterns occur frequently, then they become discrete stable states for the network.

And the network will converge on them given any similar patterns...

Emergent Property: Attractors (Discrete Stable States)
(Hopfield, 1982; 1984)

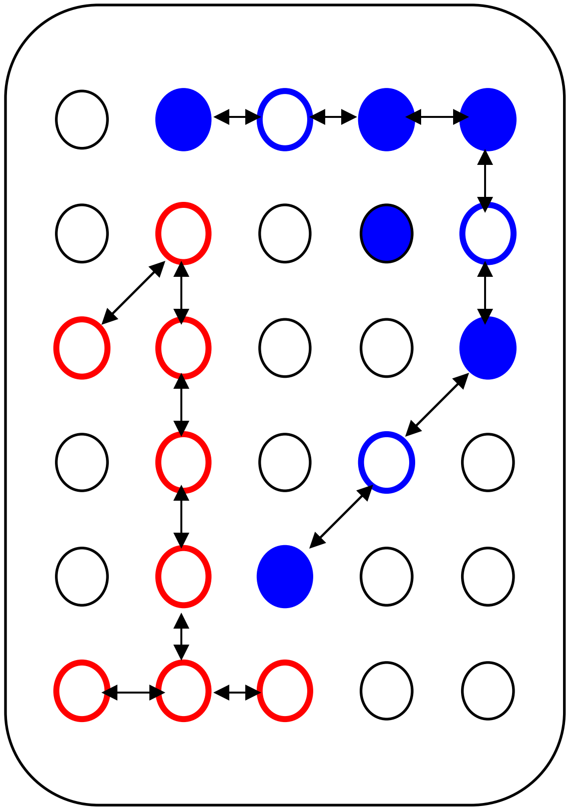


If these patterns occur frequently, then they become discrete stable states for the network.

And the network will converge on them given any similar patterns...

Emergent Property: Attractors (Discrete Stable States)

(Hopfield, 1982; 1984)

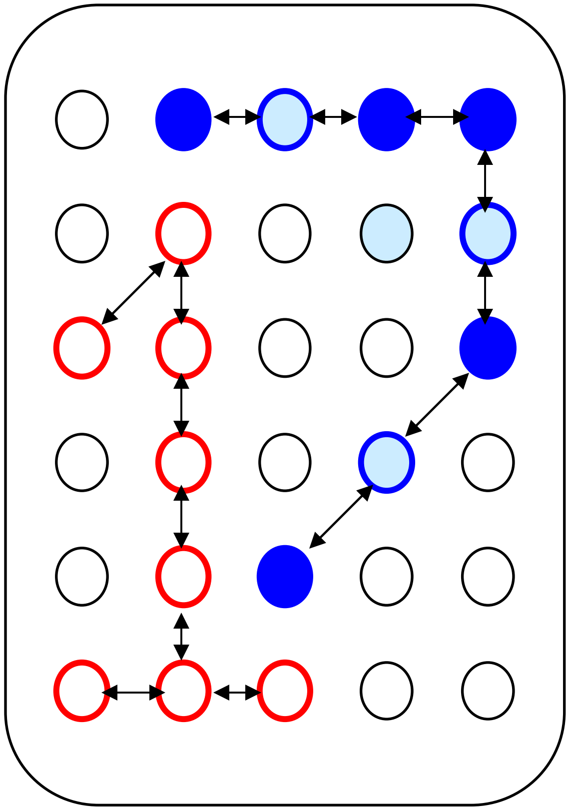


If these patterns occur frequently, then they become discrete stable states for the network.

And the network will converge on them given any similar patterns...

Emergent Property: Attractors (Discrete Stable States)

(Hopfield, 1982; 1984)

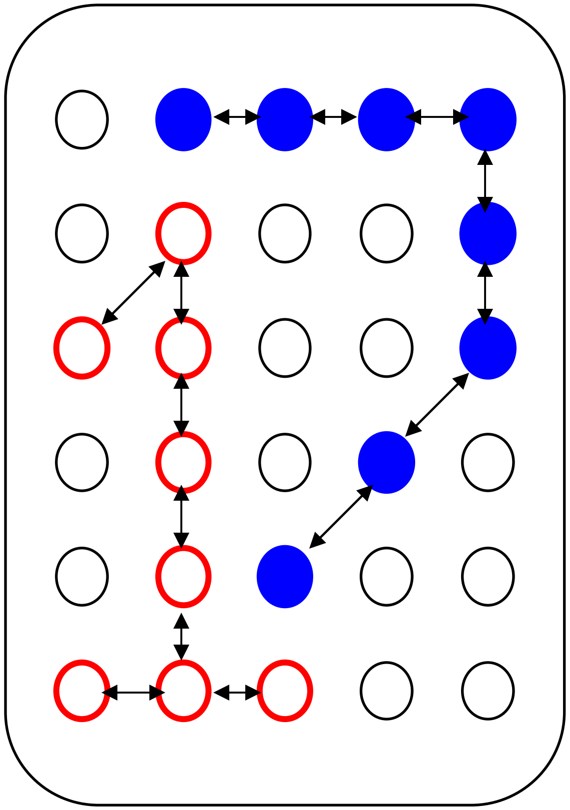


If these patterns occur frequently, then they become discrete stable states for the network.

And the network will converge on them given any similar patterns...

Emergent Property: Attractors (Discrete Stable States)

(Hopfield, 1982; 1984)



If these patterns occur frequently, then they become discrete stable states for the network.

And the network will converge on them given any similar patterns...

These “attractors” are discrete & stable like symbols.

Plan

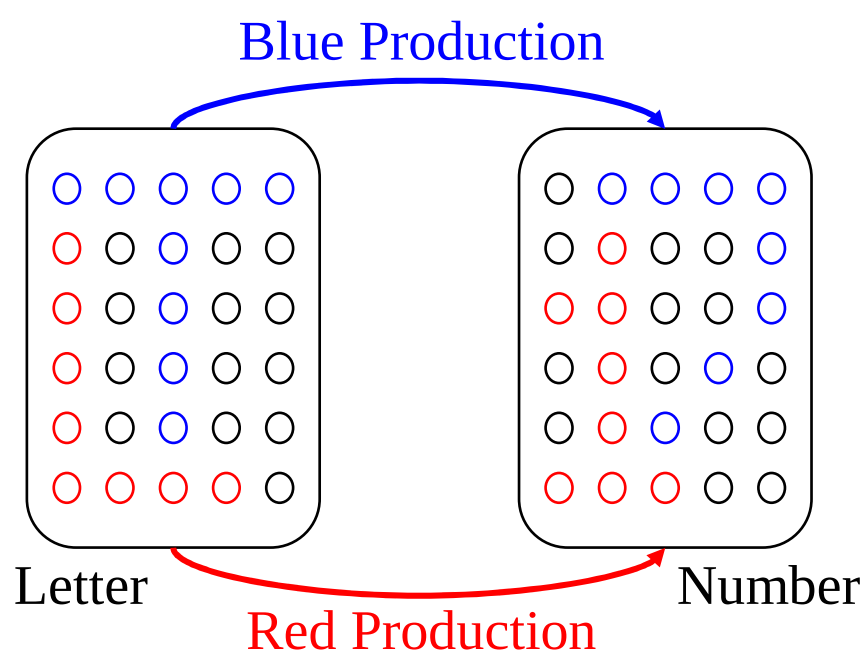
- Symbolic models of control
- A simple model of neural computation
- Mapping symbolic control onto neural nets
- Network model of Tower of London
  - Intact behavior
  - Damaged behavior

Mapping Productions Onto Attractors

- Attributes = attractor nets/layers
- Values = attractor patterns in the specified layer
- Productions = associations between layers

IF
Letter T
THEN
Number 7

IF
Letter L
THEN
Number 1

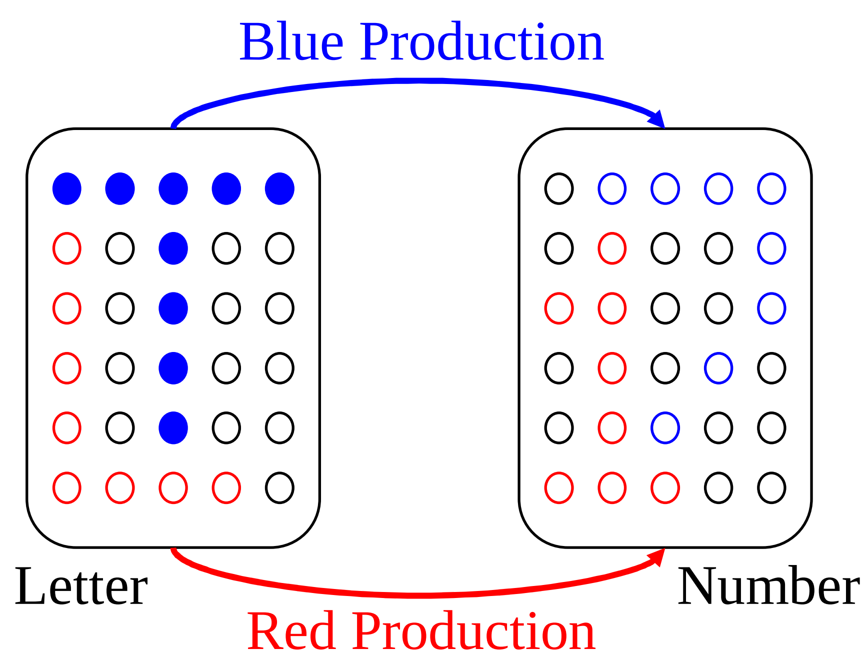


Mapping Productions Onto Attractors

- Attributes = attractor nets/layers
- Values = attractor patterns in the specified layer
- Productions = associations between layers

IF
  Letter T
THEN
  Number 7

IF
  Letter L
THEN
  Number 1

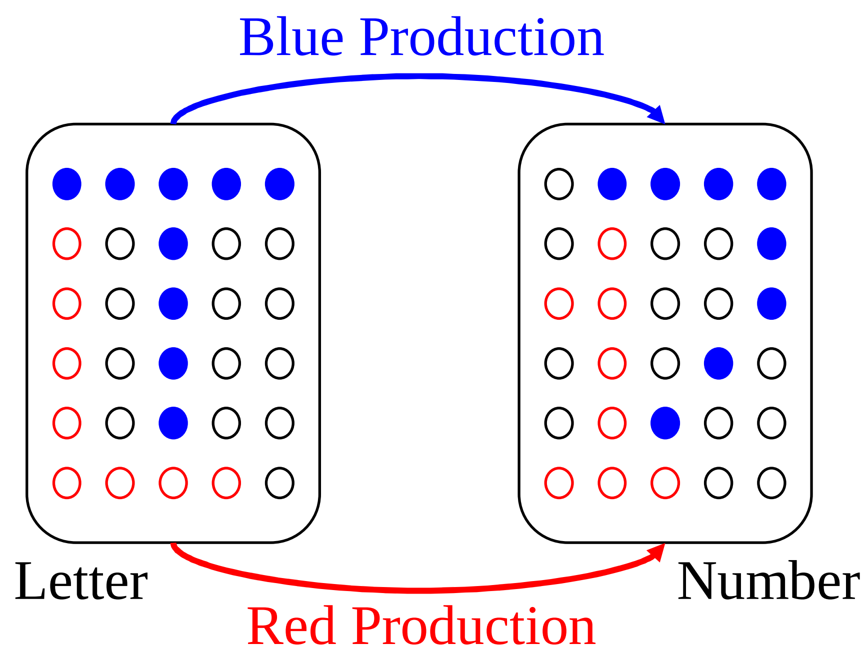


Mapping Productions Onto Attractors

- Attributes = attractor nets/layers
- Values = attractor patterns in the specified layer
- Productions = associations between layers

IF
Letter T
THEN
Number 7

IF
Letter L
THEN
Number 1

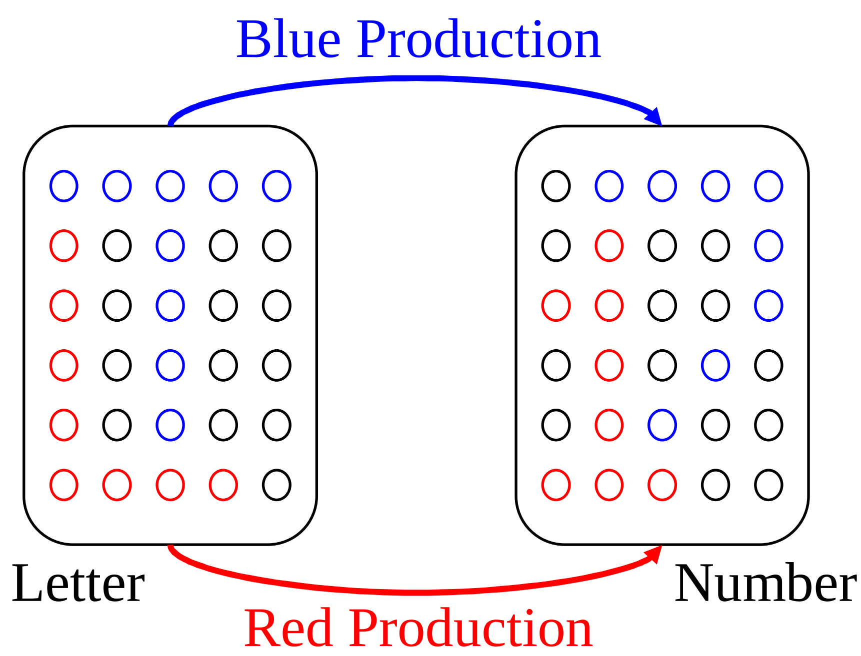


Mapping Productions Onto Attractors

- Attributes = attractor nets/layers
- Values = attractor patterns in the specified layer
- Productions = associations between layers

IF
  Letter T
THEN
  Number 7

IF
  Letter L
THEN
  Number 1

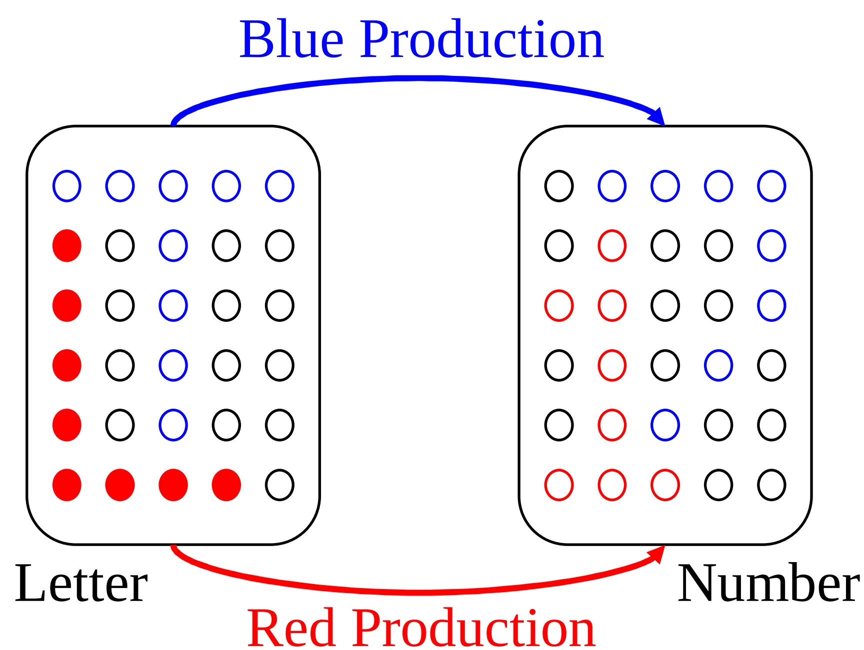


Mapping Productions Onto Attractors

- Attributes = attractor nets/layers
- Values = attractor patterns in the specified layer
- Productions = associations between layers

IF
  Letter T
THEN
  Number 7

IF
  Letter L
THEN
  Number 1

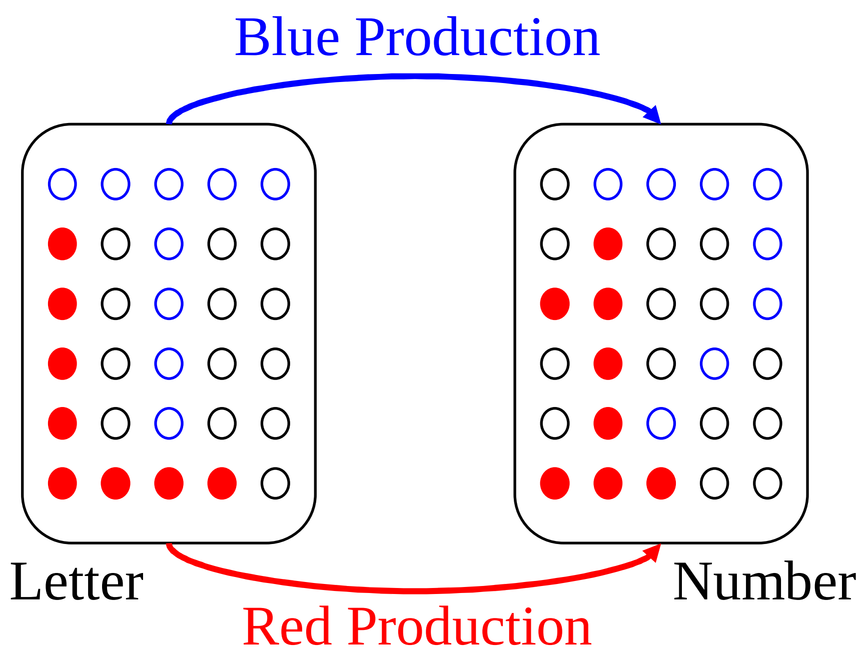


Mapping Productions Onto Attractors

- Attributes = attractor nets/layers
- Values = attractor patterns in the specified layer
- Productions = associations between layers

IF
  Letter T
THEN
  Number 7

IF
  Letter L
THEN
  Number 1

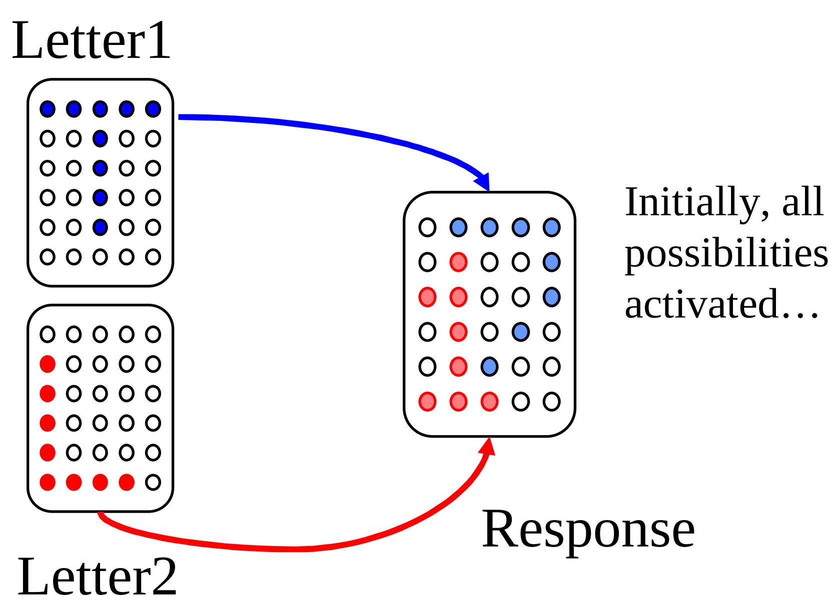


Goal-Driven Control → Attractors

- Goals bias competition among attractors
  – A kind of conflict resolution

IF
Letter1 T
THEN
Response 7

IF
Letter2 L
THEN
Response 1

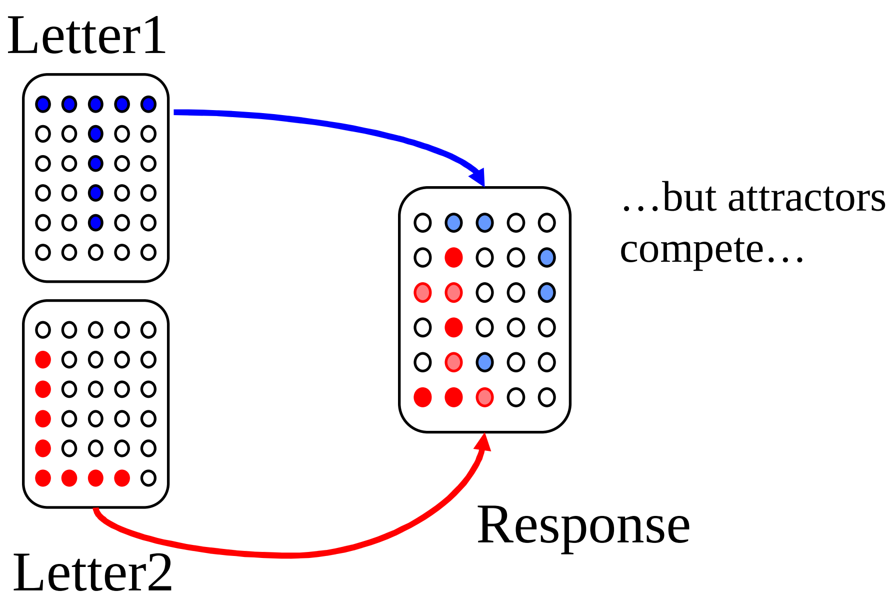


Goal-Driven Control → Attractors

- Goals bias competition among attractors
  - A kind of conflict resolution

IF
Letter1 T
THEN
Response 7

IF
Letter2 L
THEN
Response 1

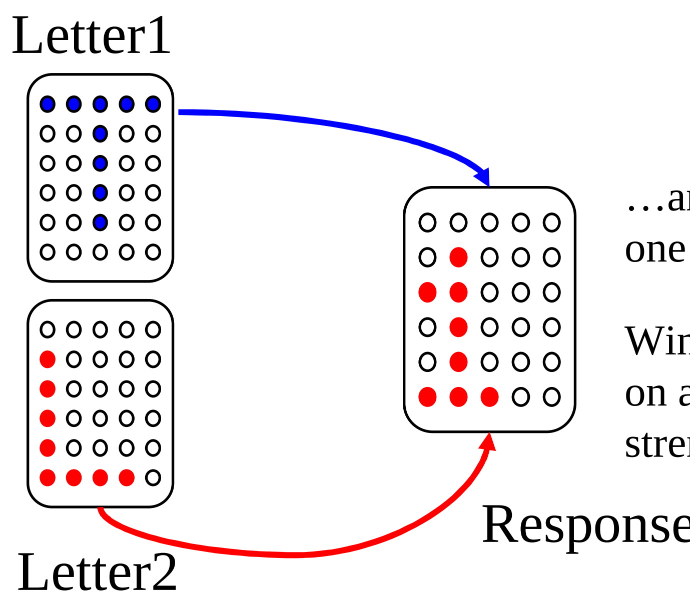


Goal-Driven Control → Attractors

- Goals bias competition among attractors
  - A kind of conflict resolution

IF
  Letter1 T
THEN
  Response 7

IF
  Letter2 L
THEN
  Response 1



…and eventually one wins.

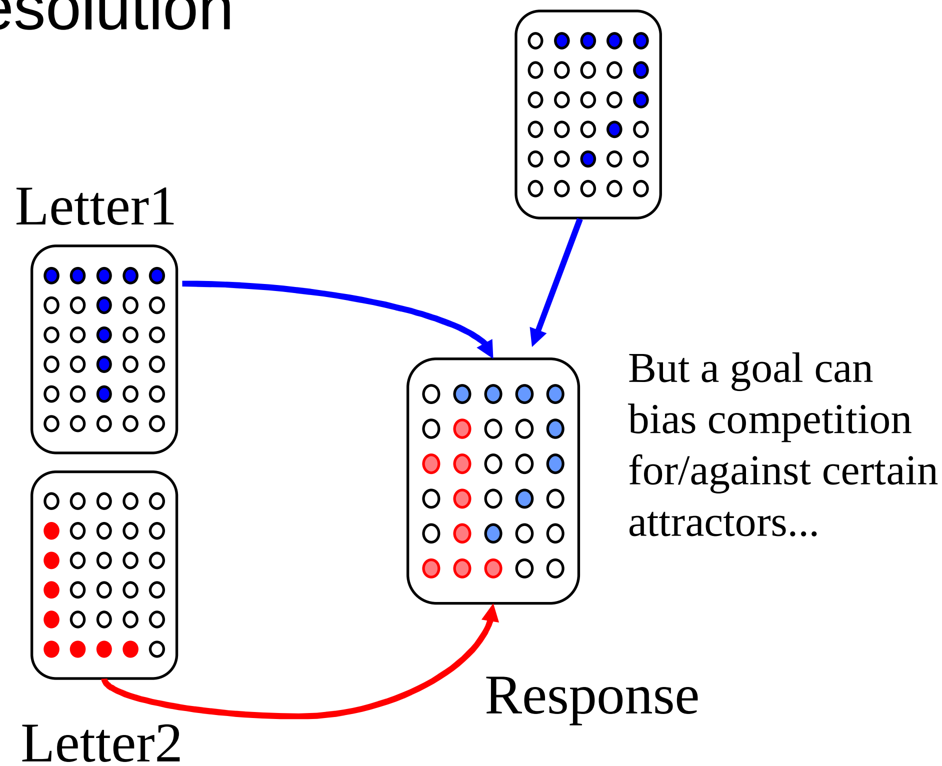
Winner depends on association strengths & noise

Goal-Driven Control → Attractors

- Goals bias competition among attractors
  - A kind of conflict resolution

IF
  Letter1 T
THEN
  Response 7

IF
  Letter2 L
THEN
  Response 1

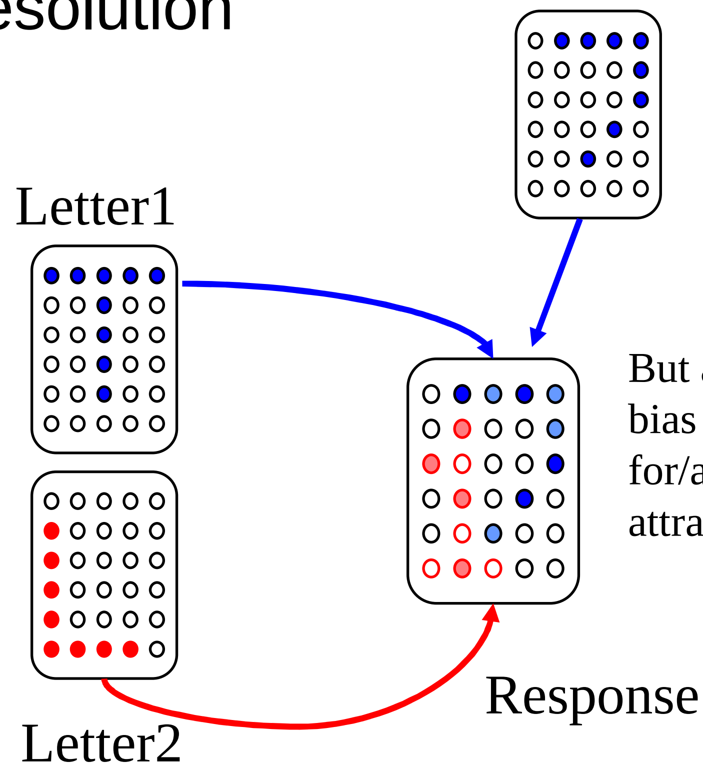


Goal-Driven Control → Attractors

- Goals bias competition among attractors
  - A kind of conflict resolution

IF
    Letter1 T
THEN
    Response 7

IF
    Letter2 L
THEN
    Response 1



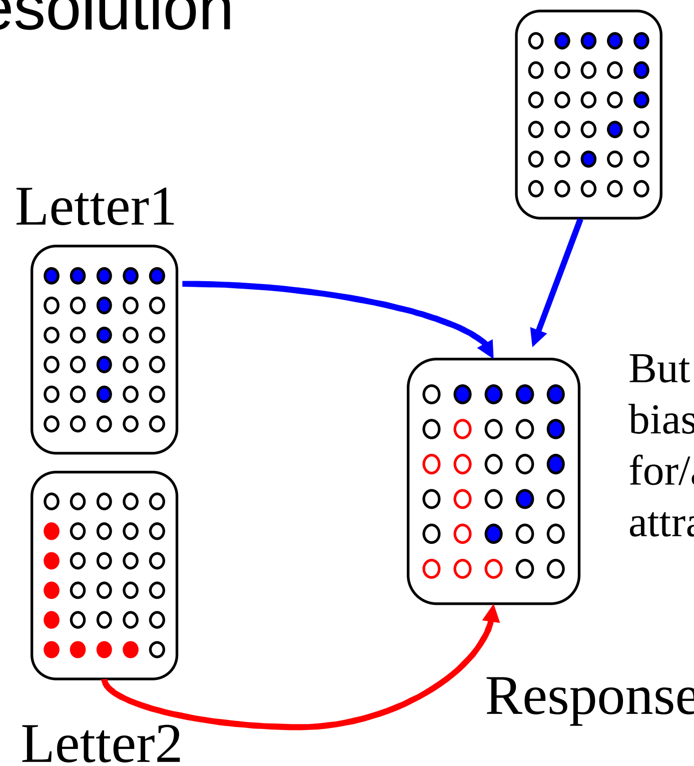
But a goal can bias competition for/against certain attractors...

Goal-Driven Control → Attractors

- Goals bias competition among attractors
  - A kind of conflict resolution

IF
    Letter1 T
THEN
    Response 7

IF
    Letter2 L
THEN
    Response 1



But a goal can bias competition for/against certain attractors...

Mapping Productions To Attractors

- Attributes = attractor nets/layers
- Values = attractor patterns in the specified layer
- Productions = associations between layers

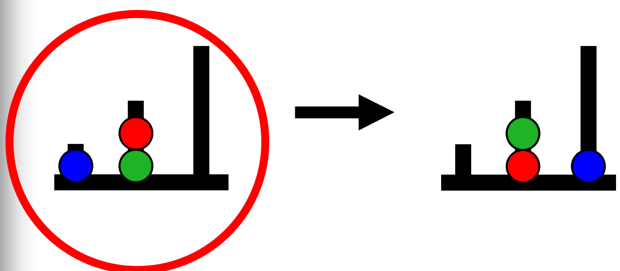
- We've implemented this mapping in Lisp/Perl:
  - Input: Simplified production rules
  - Output: Matlab code that implements attractor nets that (often!) behave like the production system

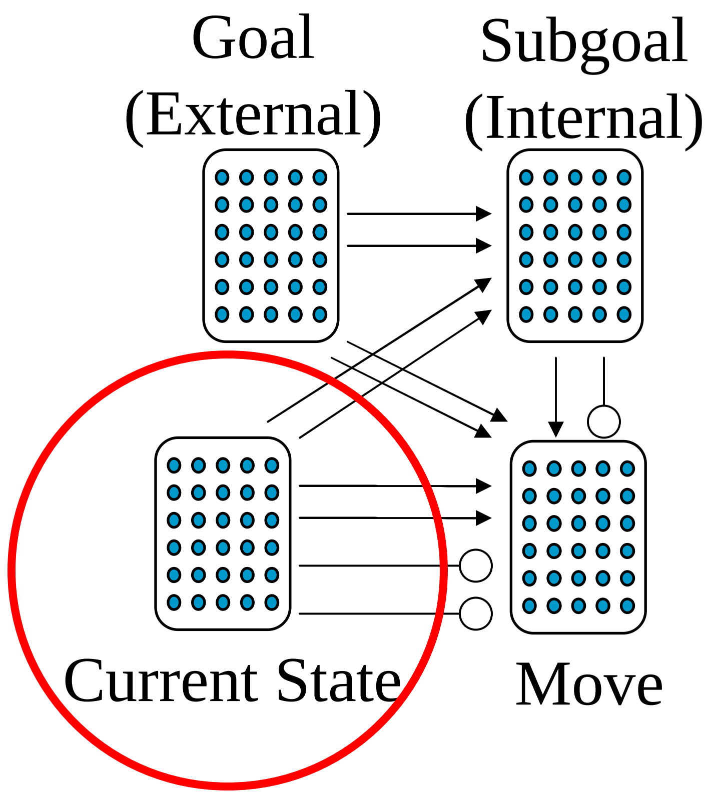
- Can thus use the attractor architecture for some higher cognitive tasks

Plan

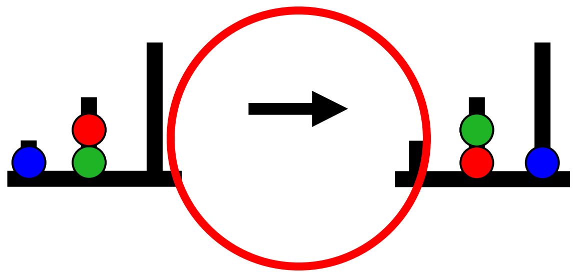
- Symbolic models of control
- A simple model of neural computation
- Mapping symbolic control onto neural nets
- Network model of Tower of London
  – Intact behavior
  – Damaged behavior

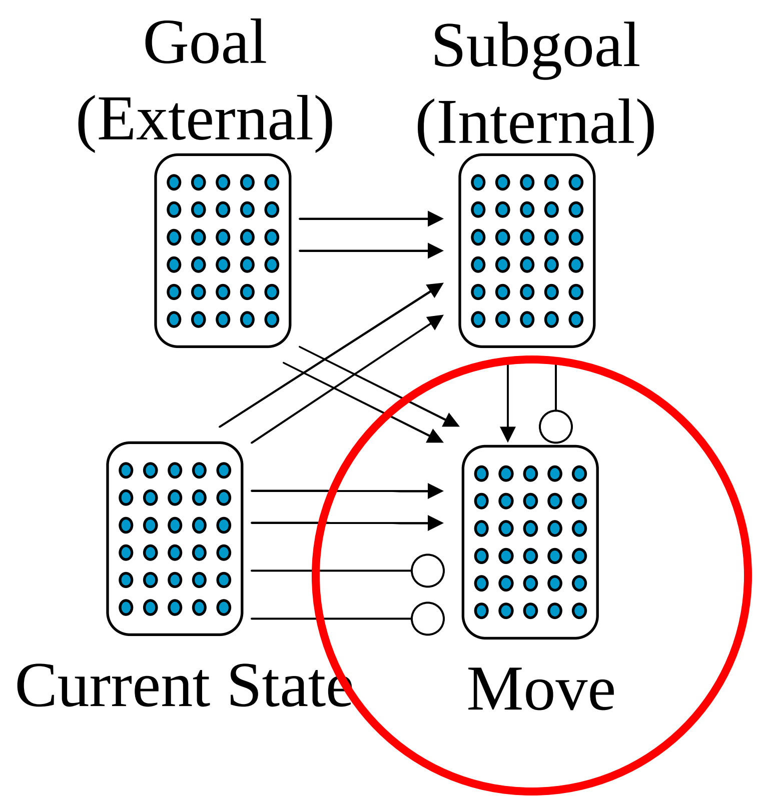
Modeling Tower of London



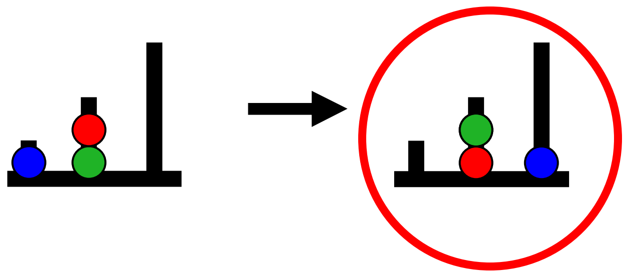


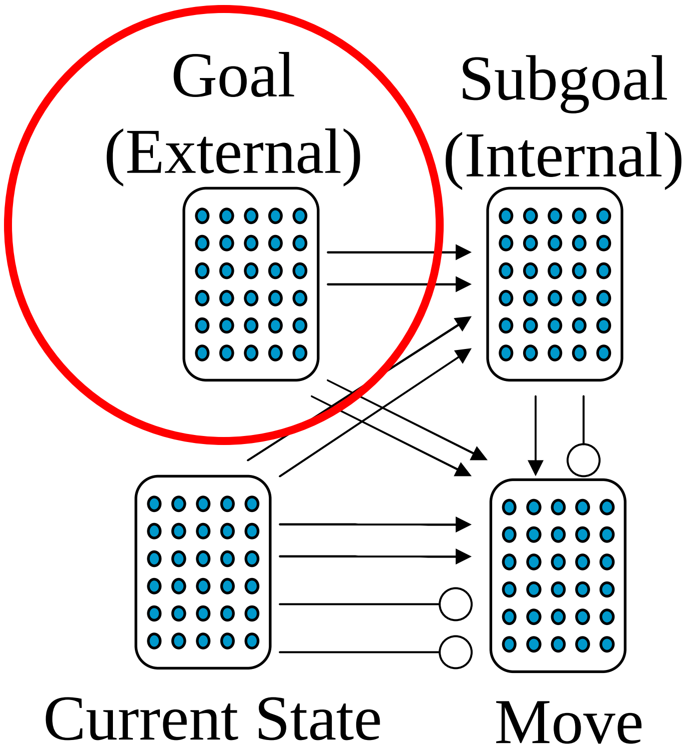
Modeling Tower of London



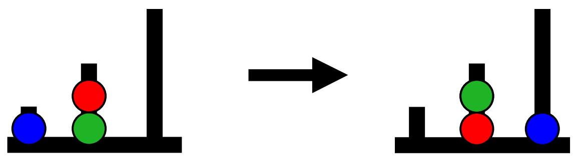


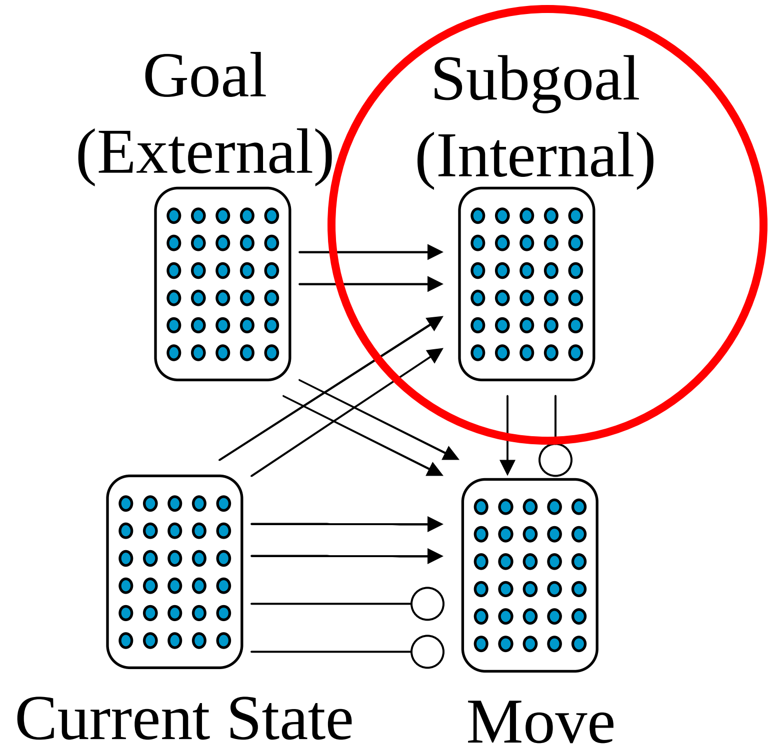
Modeling Tower of London



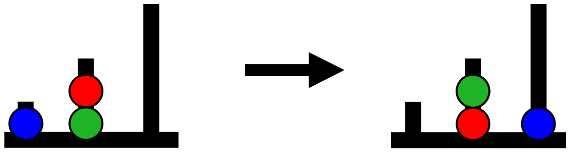


Modeling Tower of London



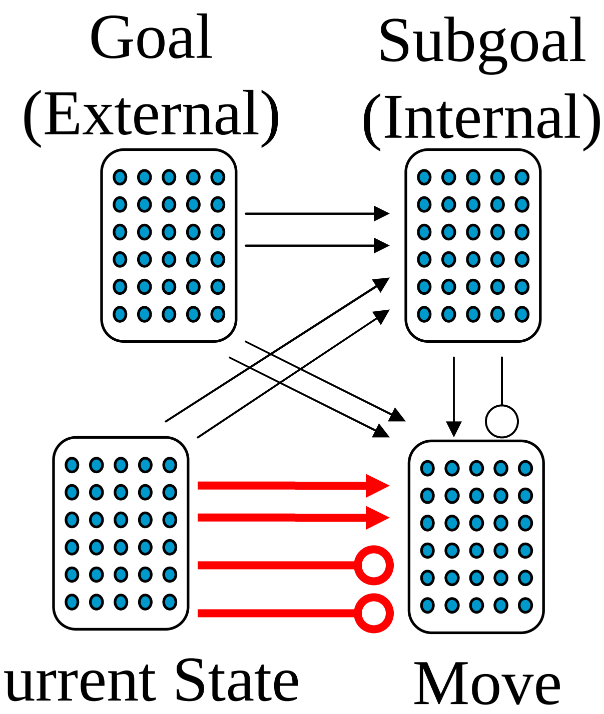


Modeling Tower of London

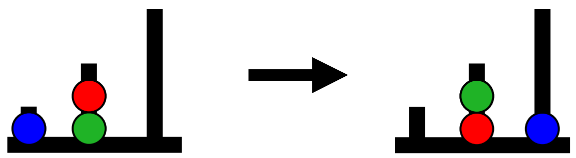


Knowledge

Legal Moves to consider

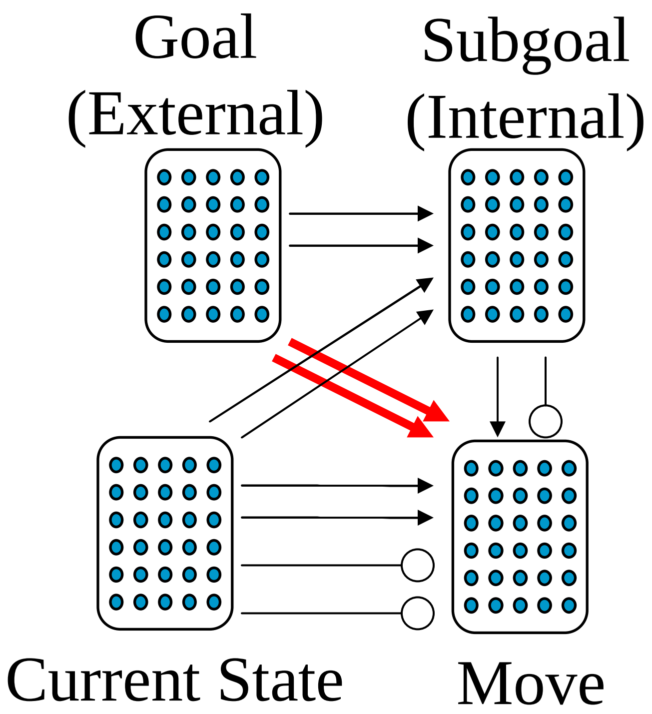


Modeling Tower of London

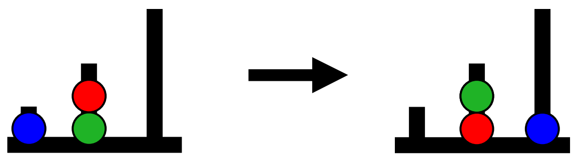


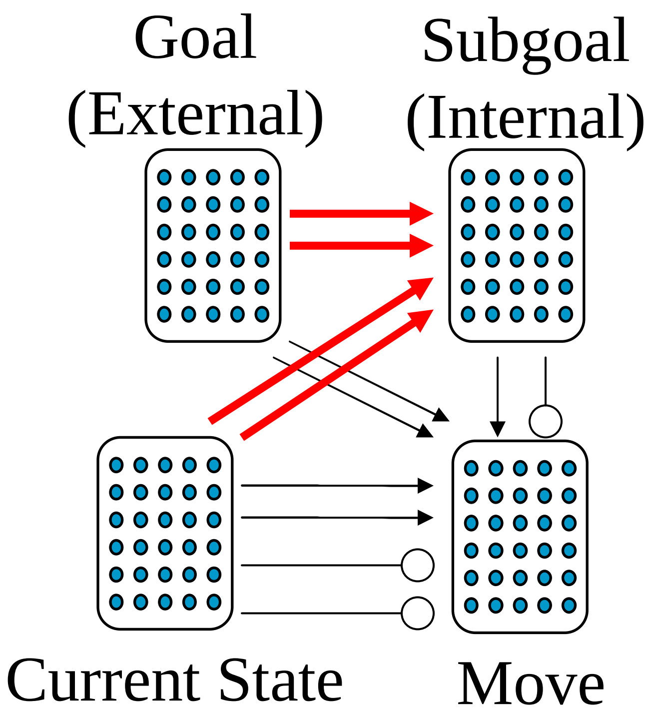
<u>Knowledge</u>

Legal Moves to consider
Try to achieve goals



Modeling Tower of London

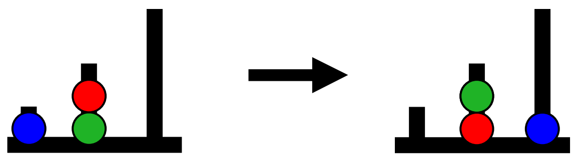




Knowledge

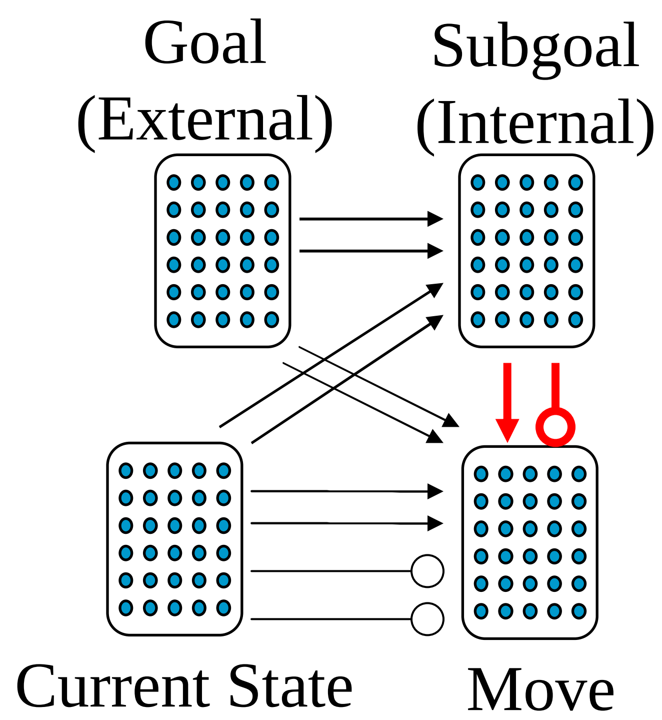
Legal Moves to consider
Try to achieve goals
Set subgoals to remove blockers

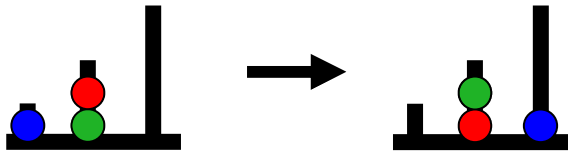
Modeling Tower of London



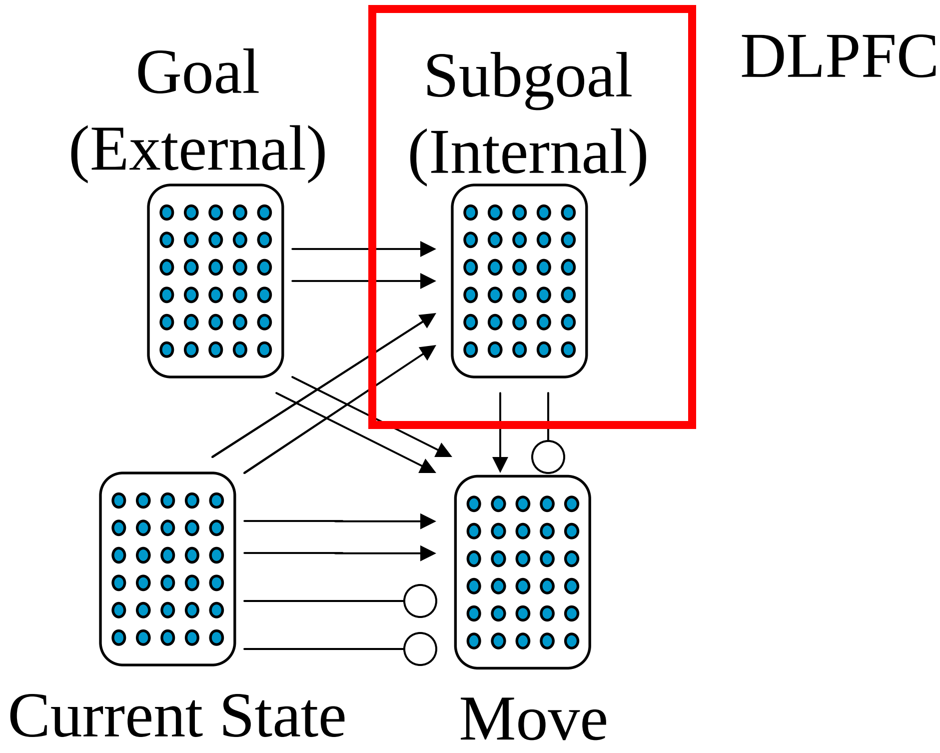
Knowledge

Legal Moves to consider
Try to achieve goals
Set subgoals to remove blockers
Try to achieve subgoals





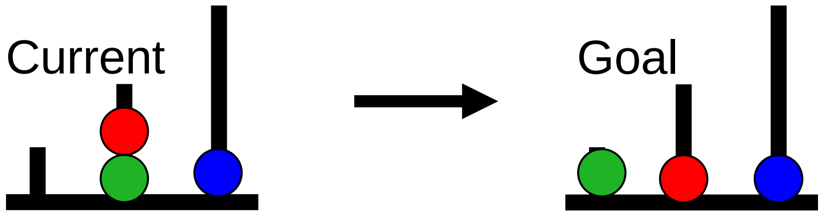
The Role of DLPFC

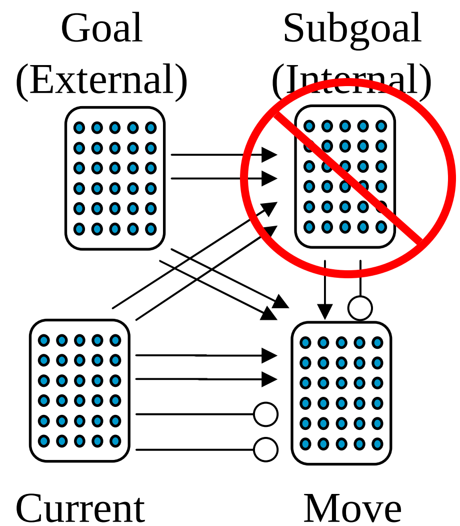


Hypothesis: DLPFC represents internal subgoals

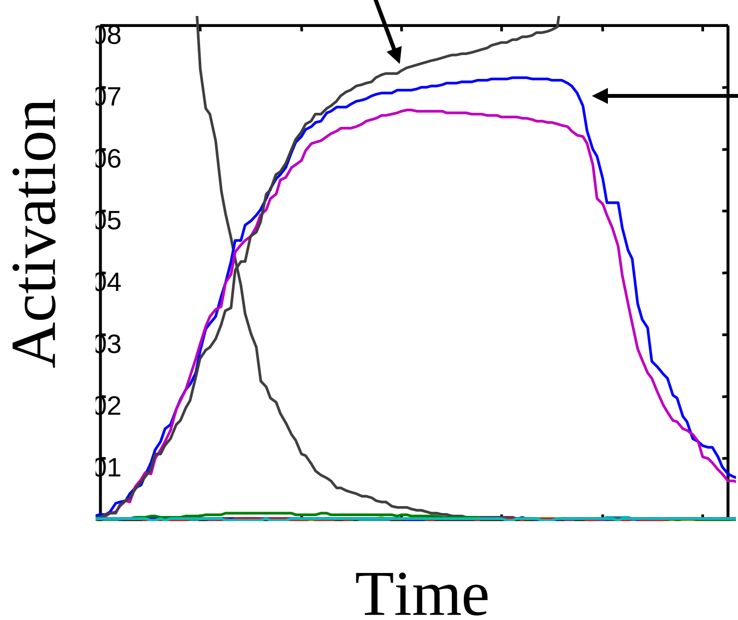
- Modulates competition among posterior attractors
- Note: Not needed for externally provided goals

The Function of DLPFC Subgoals





“Incorrect” legal move
(e.g., Red to peg 1) wins



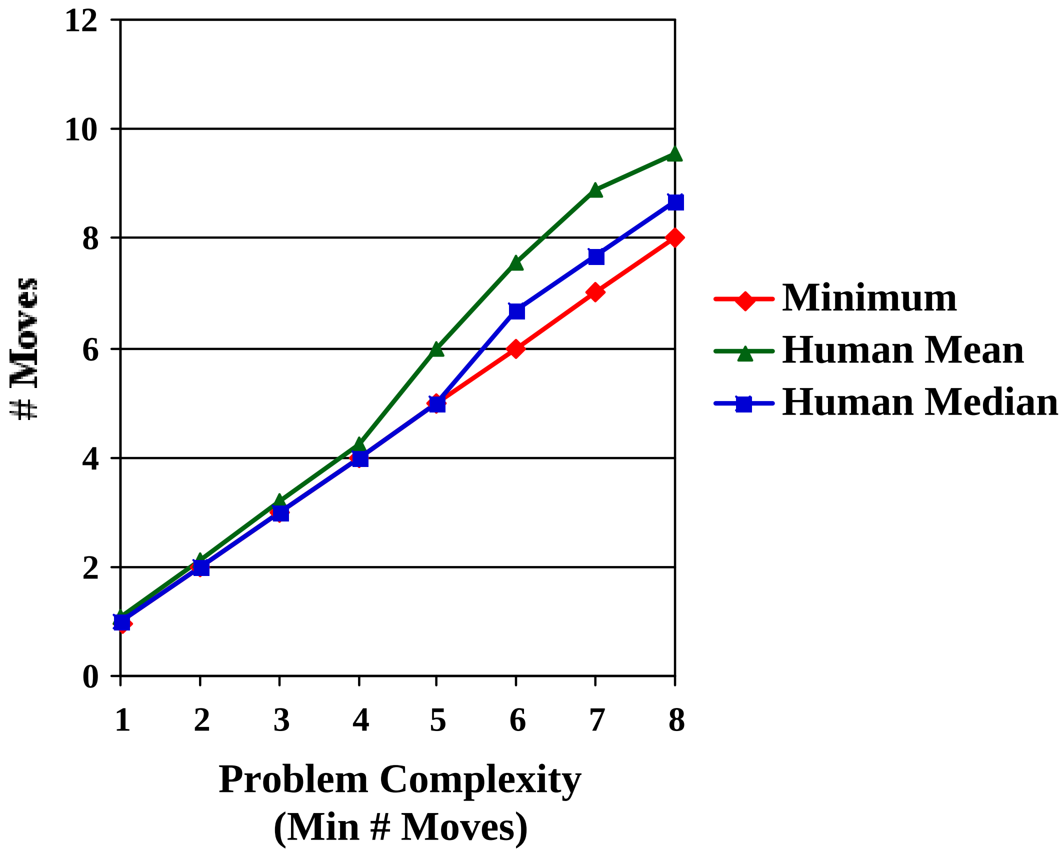
“Correct” subgoal move
(e.g., Red to peg 3) loses

When subgoal net damaged, noise can overwhelm weak input from subgoal leading wrong move to be chosen.

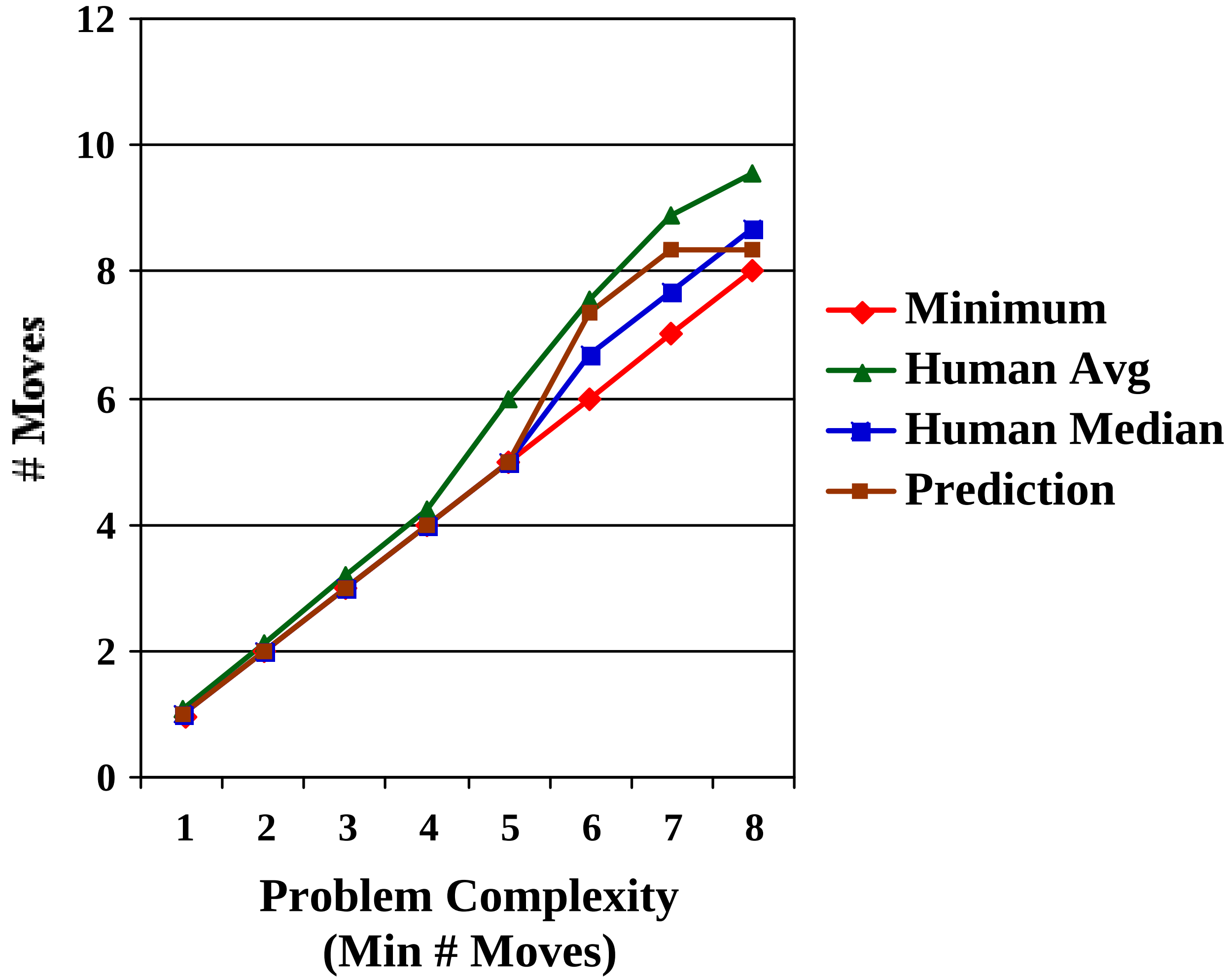
Human Data

42 undergraduates (Intact?)

18 TOL problems varying in minimum number of moves



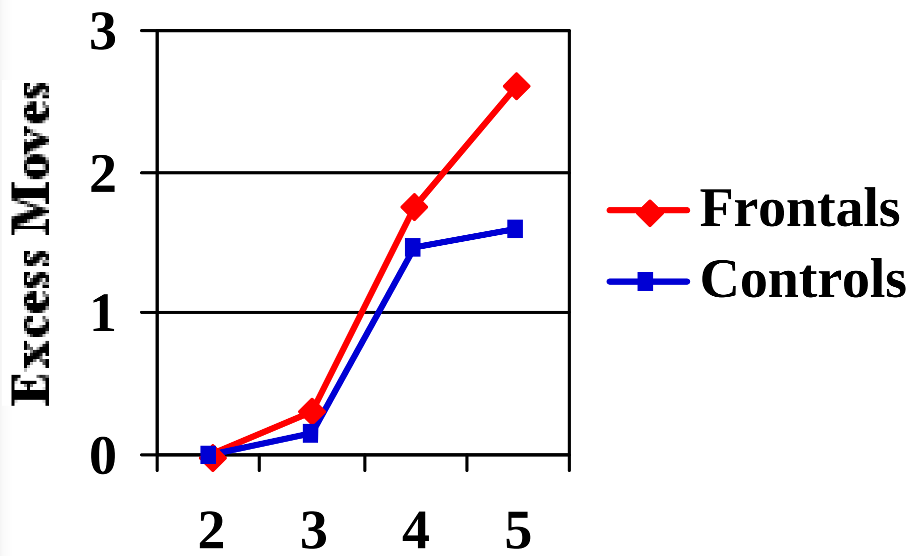
Intact Simulation



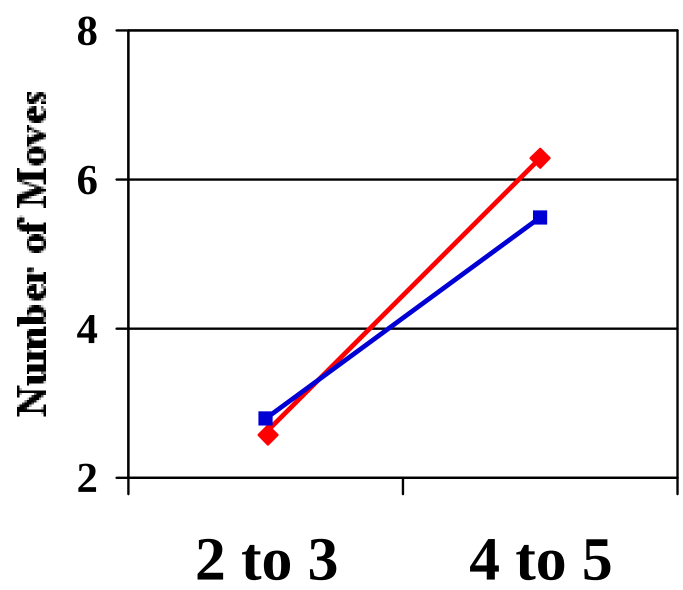
Prefrontal Data

Group x Difficulty interaction:
Prefrontal deficit is more apparent on harder problems

Owen et al. (1990)

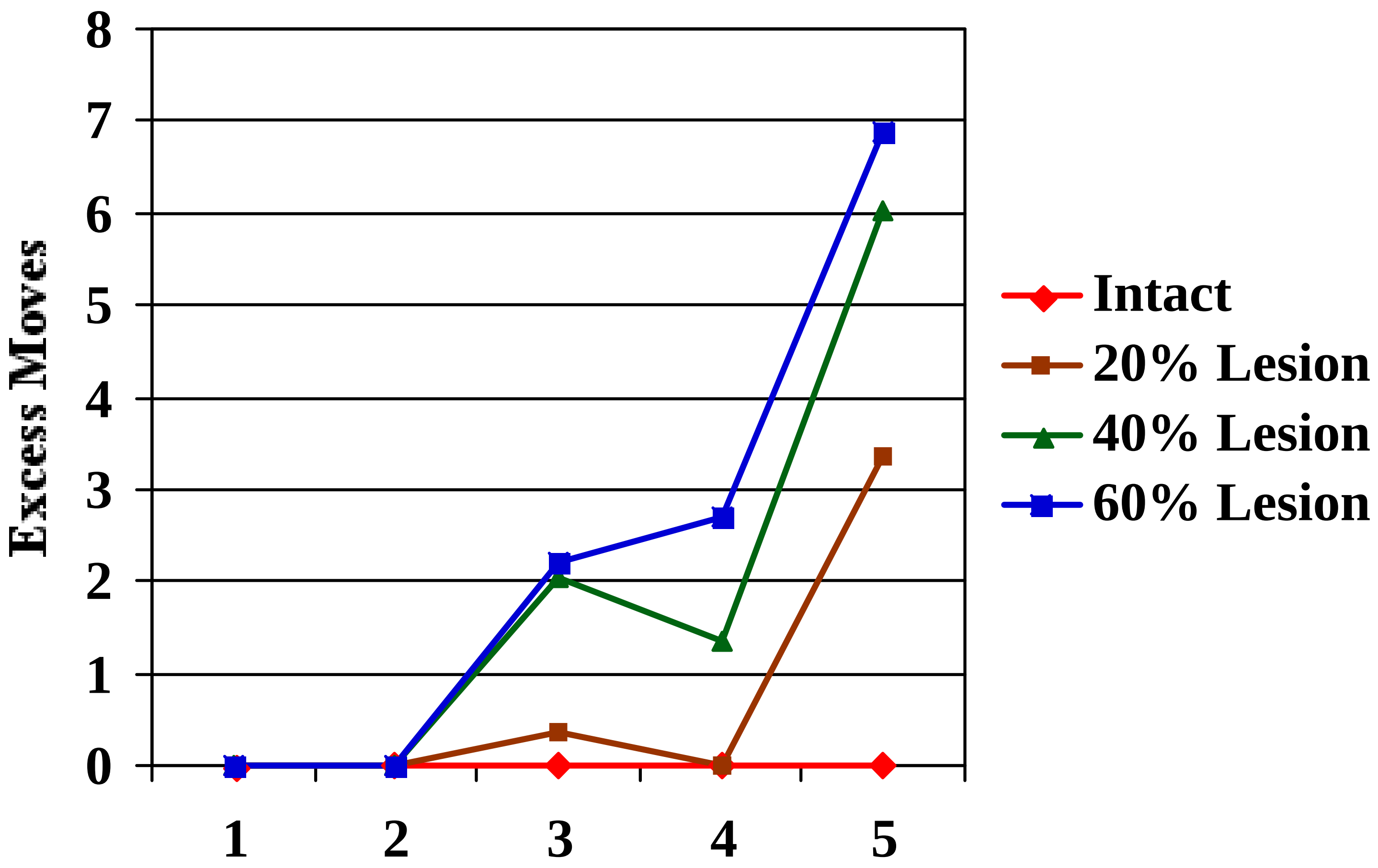


Carlin et al. (2000)

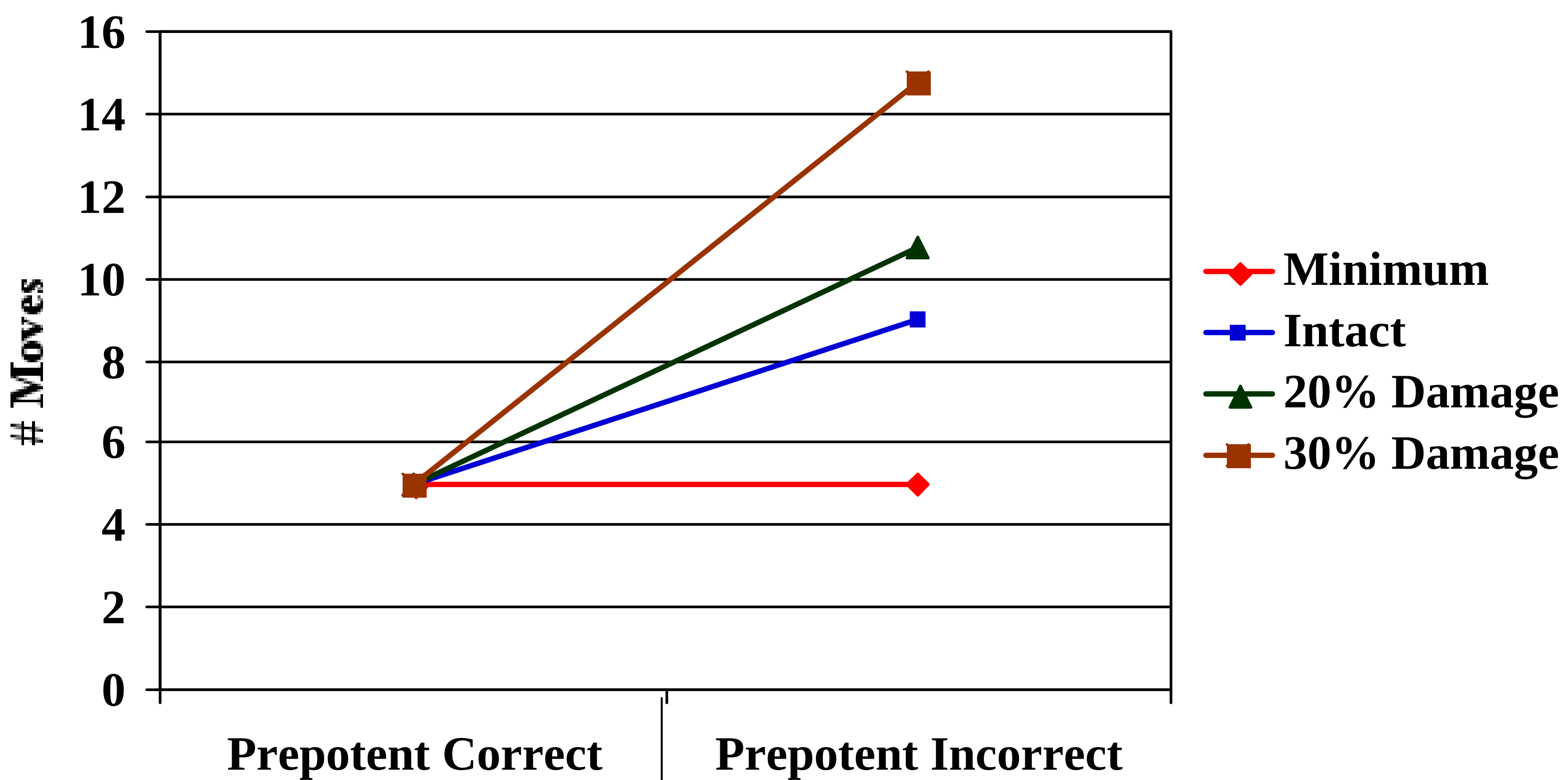


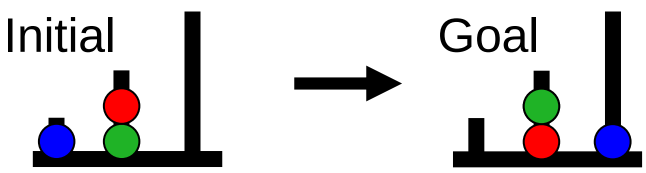
Problem Complexity
(Minimum Number of Moves)

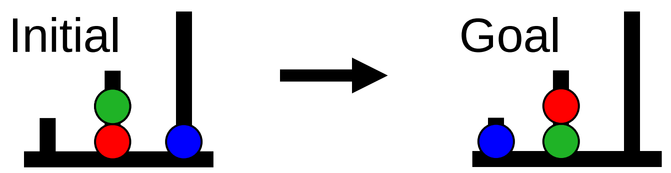
Prefrontal Simulations



Problems with Prepotent Responses







Issues Raised by the Mapping

- A number of features of production systems DON'T map easily:
  - Variables
  - Independence/modularity of different productions
  - All-or-none matching

- Possible responses:
  - Neural nets lack critical functionality for modeling cognition
    - Need to work on increasing their functionality
  - Production systems have too much functionality
    - Might use less powerful production systems that map more naturally
  - Both are probably reasonable

Summary

1. Natural & explicit mapping from goal-driven production systems onto neural computation
   – Makes it possible to build plausible neural models of complex problem solving

2. Mapping leads to explicit hypothesis about the role of DLPFC in problem solving:
   – Represents internally generated subgoals that modulate among choices

3. Applying to TOL accurately simulates human behavior
   – Intact model simulates normals, even on hardest problems
   – Lesioning subgoal net simulates prefrontal deficits